
150 ptas.
mi computer
23
CURSO PRACTICO DEL ORDENADOR PERSONAL,
EL MICRO Y EL MINIORDENADOR
441 Juegos de guerra
444 Equipos de herramientas
446 Reconocimiento de voz
448 Lenguaje máquina
450 Sinclair QL
452 Sonido y luz
454 Criptografía
456 Programación Basic
460 Pioneros de la informática
Code Cracking
Editorial Delta, S.A.

MI COMPUTER CURSO PRACTICO
DEL ORDENADOR PERSONAL, EL MICRO Y EL MINIORDENADOR

Publicado por Editorial Delta, S.A., Barcelona, y comercializado en exclusiva por Distribuidora Olimpia, S.A., Barcelona

Volumen II - Fascículo 23

Director: José Mas Godayol
Director editorial: Gerardo Romero
Jefe de redacción: Pablo Parra
Coordinación editorial: Jaime Mardones
Asesor técnico: Jesús Nebra

Redactores y colaboradores: G. Jefferson, R. Ford, S. Tarditti, A. Cuevas
Para la edición inglesa: R. Pawson (editor), D. Tebbutt (consultant editor), C. Cooper (executive editor), D. Whelan (art editor), Bunch Partworks Ltd. (proyecto y realización)

Realización gráfica: Luis F. Balaguer

Redacción y administración:
Paseo de Gracia, 88, 5.°, Barcelona-8
Tels. (93) 215 10 32 / (93) 215 10 50 - Télex 97848 EDLTE

MI COMPUTER, *Curso práctico del ordenador personal, el micro y el miniordenador,* se publica en forma de 96 fascículos de aparición semanal, encuadernables en ocho volúmenes. Cada fascículo consta de 20 páginas interiores y sus correspondientes cubiertas. Con el fascículo que completa cada uno de los volúmenes, se ponen a la venta las tapas para su encuadernación.
El editor se reserva el derecho de modificar el precio de venta del fascículo en el transcurso de la obra, si las circunstancias del mercado así lo exigieran.

ISBN: 84-85822-83-8 (fascículo) 84-85822-90-0 (tomo 2)
84-85822-82-X (obra completa)
Depósito Legal: B. 52/1984

Fotocomposición: Tecfa, S.A., Pedro IV, 160, Barcelona-5
Impresión: Cayfosa, Santa Perpètua de Mogoda (Barcelona) 208406
Impreso en España - Printed in Spain - Junio 1984

Editorial Delta, S.A., garantiza la publicación de todos los fascículos que componen esta obra.

Distribuye para España: Marco Ibérica, Distribución de Ediciones, S.A., Carretera de Irún, km 13,350. Variante de Fuencarral, Madrid-34.
Distribuye para Argentina: Viscontea Distribuidora, S.C.A., La Rioja 1134/56, Buenos Aires.
Distribuye para Colombia: Distribuidoras Unidas, Ltda., Transversal 93, n.° 52-03, Bogotá D.E.
Distribuye para México: Distribuidora Intermex, S.A., Lucio Blanco, n.° 435, Col. San Juan Tlihuaca, Azcapotzalco, 02400, México D.F.
Distribuye para Venezuela: Distribuidora Continental, S.A., Ferrenquín a Cruz de Candelaria, 178, Caracas, y todas sus sucursales en el interior del país.

Pida a su proveedor habitual que le reserve un ejemplar de MI COMPUTER. Comprando su fascículo todas las semanas y en el mismo quiosco o librería, Vd. conseguirá un servicio más rápido, pues nos permite realizar la distribución a los puntos de venta con la mayor precisión.

Servicio de suscripciones y atrasados (sólo para España)

Las condiciones de suscripción a la obra completa (96 fascículos más las tapas, guardas y transferibles para la confección de los 8 volúmenes) son las siguientes:

a) Un pago único anticipado de 16 690 ptas. o bien 8 pagos trimestrales anticipados y consecutivos de 2 087 ptas. (sin gastos de envío).

b) Los pagos pueden hacerse efectivos mediante ingreso en la cuenta 3371872 de la Caja Postal de Ahorros y remitiendo a continuación el resguardo o su fotocopia a Distribuidora Olimpia (Paseo de Gracia, 88, 5.°, Barcelona-8), o también con talón bancario remitido a la misma dirección.

c) Se realizará un envío cada 12 semanas, compuesto de 12 fascículos y las tapas para encuadernarlos.

Los fascículos atrasados pueden adquirirse en el quiosco o librería habitual. También pueden recibirse por correo, con incremento del coste de envío, haciendo llegar su importe a Distribuidora Olimpia, en la forma establecida en el apartado b).

Para cualquier aclaración, telefonear al (93) 215 75 21.

No se efectúan envíos contra reembolso.

Editorial Delta, S.A.

Guerra y paz

En este escenario ficticio, se teme una incursión de bombarderos sobre el campo de aviación de Luton, así que enviamos una escuadrilla británica para patrullar por las inmediaciones

LUTON

Para nuestra sorpresa, las dos formaciones de aparatos alemanes se lanzan sobre Londres, donde los interceptamos con una fuerza aérea más pequeña, que les produce importantes bajas

LONDRES

DUNKIRK

De regreso a Francia, los alemanes bombardean con éxito la estación de radar de Dunkirk, en Kent, desprovista, por descuido, de cazas británicos

DIEPPE

CHERBURGO

Kevin Jones

Muchos juegos ponen a prueba la aptitud como estratega militar del usuario, simulando batallas históricas o hipotéticas

En la actualidad, los generales modernos conceden una gran importancia a los juegos de guerra donde planificar respuestas adecuadas a un supuesto ataque o conspiración. Para jugar a estos sofisticados juegos se han ideado complicados sistemas de hardware y software combinando todos los aspectos conocidos de un conflicto potencial, como el despliegue inicial de las fuerzas aliadas y enemigas, las condiciones de aprovisionamiento, la retaguardia, etc. El sistema incluye hasta las condiciones climatológicas adversas, los cambios de táctica del enemigo, los efectos de las actividades de la quinta columna, y cualquier otra variable que tenga que ver con el éxito de una operación militar. Una de las principales tareas encomendadas al sistema

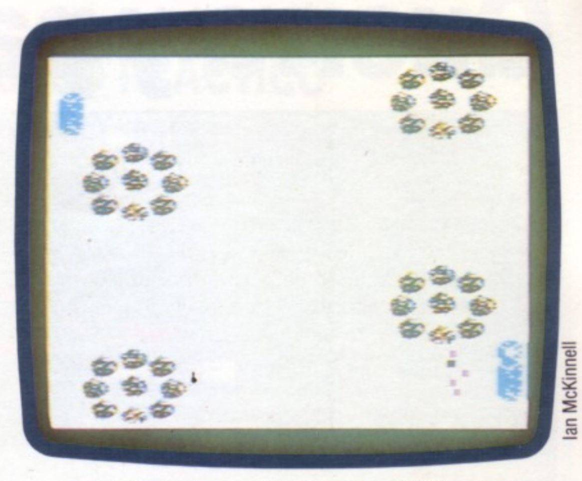


Ian McKinnell

La Batalla de Inglaterra
Fighter Command (ideado por Strategic Simulations Inc. para el Apple) es uno de los típicos juegos de estrategia bélica para microordenadores. Antes de comenzar, el jugador debe elegir entre una gama de opciones muy amplia, inclusive el tipo de avión a utilizar y las condiciones meteorológicas. La acción se visualiza en forma de símbolos que se mueven sobre el mapa, junto con una información adicional en forma de texto. El paquete y la documentación incluyen un mapa impreso con piezas de cartón para una referencia visual suplementaria

Tactical Armour Command
El TAC (de Avalon Hill) es utilizable por los ordenadores Atari, Apple, IBM PC y Commodore 64. Simula un combate de acorazados durante la segunda guerra mundial. El TAC es para uno o dos jugadores, quienes seleccionan entre cinco escenarios diferentes y manipulan fuerzas británicas, norteamericanas, rusas y alemanas





Ian McKinnell

NORAD de defensa por radar de las montañas Cheyenne (Wyoming) —descrito en la película *Juegos de guerra (War games)*— es la de determinar, actualizar y calibrar de forma continua la potencia relativa de Estados Unidos y de la Unión Soviética y asesorar la preparación de una respuesta ante cualquier eventualidad.

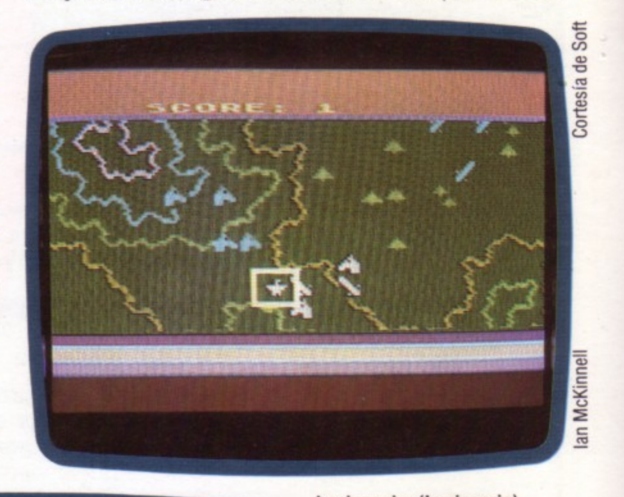
Por supuesto, los juegos de guerra para generales amateurs son menos sutiles. Con el fin de crear el adecuado nivel de complejidad, el jugador de guerra ha debido remitirse a hojas con tablas, voluminosos libros de reglas y un gran número de dados. La indecible fatiga que supone la práctica de juegos de guerra los ha ido restringiendo a un grupo de entusiastas no tan reducido. Sin embargo, con el advenimiento del ordenador personal y la disponibilidad de programas para juegos de guerra, aquel tedioso "trabajo de Estado Mayor" ha desaparecido y ha dejado en su lugar un absorbente juego que ofrece tanto incitación como desafío, al igual que cualquier otro tipo de juegos para ordenador de los que se encuentran habitualmente en el mercado.

Hay una riquísima variedad de juegos. Se puede recrear o simular prácticamente cualquier tipo de operación militar desde los tiempos de la Grecia antigua hasta una teórica confrontación entre la OTAN y el Pacto de Varsovia allá por el día de sanjamás. Se pueden librar batallas aéreas, navales, guerras de galaxias e incluso guerras entre míticos imperios. El panorama es ilimitado.

Los juegos históricos ofrecen la posibilidad de descubrir por qué Napoleón se equivocó en Waterloo o de intentar ser más listos que Hitler desbaratando su intento de invadir Rusia en 1941. Los juegos cósmicos desafían aún más nuestra inventiva. No sólo podemos maniobrar una escuadra de naves interestelares a través de las galaxias, sino también especificar la clase de naves que deseamos. Por supuesto, bajo determinados compromisos. A cambio de mayor velocidad puede que debamos sacrificar armamento, y un refuerzo antibalas de la pantalla quizá se consiga a cambio de menor aprovisionamiento de combustible. El juego consiste en optar por el compromiso que mejor se adapte al estilo con que libramos una batalla.

A diferencia de sus equivalentes convencionales, los juegos de estrategia por ordenador no exigen ninguna pericia ni conocimientos previos, y la

Cortesía de Soft



Ian McKinnell





Legionnaire (Legionario)
Esta simulación del arte de la guerra entre las fuerzas de César (usted) y los bárbaros (el ordenador: un Apple o un Atari) se juega en tiempo real. La infantería, la caballería y las otras fuerzas se representan por medio de símbolos, que se pueden seleccionar y desplazar mediante el cursor (cuadrado blanco) controlado por una palanca de mando. El juego es una producción de Avalon Hill

mayoría de ellos viene con breves notas y pistas para los principiantes. No obstante, conviene saber que algunos juegos se clasifican como de iniciación, intermedios o avanzados. Si usted se dispone a practicar juegos de estrategia, lo mejor que puede hacer es comenzar por el nivel de iniciación, y asimilar los conceptos básicos de los juegos de guerra y las simulaciones estratégicas antes de probar con los juegos de niveles más avanzados.

El formato de los juegos varía según los diferentes tipos de operaciones militares representadas. Por lo general se practican sobre grandes mapas. Y si son tan extensos que no se pueden visualizar en una sola pantalla, entonces ésta suele actuar como una ventana móvil (dirigida por una palanca de mando) a través del mapa. En las figuraciones históricas, los diseñadores de juegos intentan reproducir, con la mayor fidelidad posible, el terreno sobre el cual se libró la batalla original. En *Computer Bismarck*, de SSI, la acción se desarrolla en el Atlántico Norte, lo que no supuso demasiados problemas en cuanto al diseño de los gráficos. Por el contrario, para otro juego de SSI, *Battle for Normandy* (La batalla de Normandía), el trabajo de los diseñadores fue muchísimo más difícil. No sólo el terreno general debía ser el apropiado, sino también los lugares específicos, como las playas y la costa, las ciudades, los pueblos y los ríos. En los juegos no históricos, el diseñador tiene amplio margen para hacer que el jugador se centre sobre todo en las fuerzas de que dispone, pero aun en este caso el diseñador ha de tener mucho cuidado de incluir los obstáculos y las compensaciones suficientes para evitar que el juego sea demasiado sencillo por uno u otro bando.

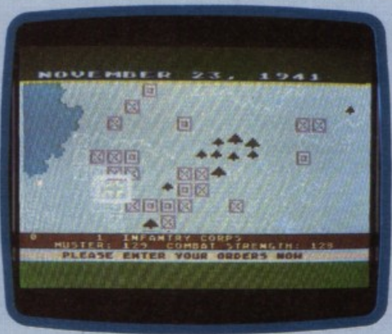
El mapa tiene también una cuadrícula superpuesta. Esta cuadrícula trocea el mapa al modo como se recuadra un tablero de ajedrez, si bien para los mapas para juegos de guerra suelen preferirse retículas hexagonales antes que cuadradas. A cada cuadrado o hexágono se le otorga un valor según el tipo de terreno que encierre. Este valor representa el grado de dificultad que tendría una unidad para alcanzar esa zona o atravesarla. El esfuerzo invertido en tales desplazamientos se traduce en una reducción de la capacidad de movimiento según el valor acordado. Cuando el margen de movimiento de la unidad equivale a cero, o es inferior al valor de la zona en la que se propone entrar, cabe la posibilidad de que en esa jugada ya no pueda avanzar más.

Por lo general el juego se divide en una cantidad de "jugadas" que representan el tiempo transcurrido, y cada jugador se asigna unos objetivos que debe alcanzar en el tiempo disponible si quiere ganar. En la mayoría de los casos no es necesario, o ni siquiera posible, alcanzar todos los objetivos fijados. De modo que la primera decisión que ha de tomar el jugador es juzgar cuáles son sus posibilidades y, en función de ellas, determinar sus prioridades estratégicas. En tales circunstancias el papel del contrincante consiste en evitar que el atacante consiga los objetivos fijados. Tampoco a él le será posible protegerlo todo, de modo que habrá de decidir cuándo abandonar defensas desesperadas, durante cuánto tiempo resistir en las plazas fuertes, y si correr o no el riesgo de lanzar contraataques para recuperar posiciones perdidas o desbaratar los preparativos de su contrario para un nuevo ataque.

El jugador se comunica con el programa a través de la representación gráfica y textual de las fuerzas bajo su mando que están en el mapa. La visualización gráfica representa la situación de una unidad determinada en el campo de batalla y la textual proporciona información relativa a la eficacia en combate de la unidad y al margen de movimiento. El jugador desplaza sus unidades señalándolas mediante un cursor o haciendo que el ordenador se las presente en rotación. Una vez escogida la unidad, se da la orden de desplazamiento. En el caso de un mapa con divisiones hexagonales, 1 enviaría la unidad hacia el norte, 2 la enviaría hacia el noreste, y así sucesivamente a los puntos de la brújula. Es cada vez mayor el número de estos juegos que funcionan a base de palancas de mando o mandos de bola, los cuales "toman" una unidad y la mueven simplemente. Para dar por terminado el desplazamiento se suele utilizar la orden FINISH o F. Aun entonces, algunos juegos ofrecen la variante de permitir que el jugador vuelva a señalar la unidad y la mueva otra vez, a menos que haya agotado todas sus posibilidades y ya no le quede margen de movimiento. Cuando se han completado todos los movimientos, el jugador se lo indica al ordenador mediante la orden EXECUTE o E. El ordenador iniciará entonces la fase de combate.

Durante la fase de combate el ordenador indicará qué unidades aliadas están en condiciones de enfrentarse con el enemigo e informará acerca de la potencia relativa de las unidades. Sobre la base de esta información, el jugador puede aceptar o rechazar las sugerencias para el combate a medida que éstas se le van ofreciendo. Una vez llevado a cabo el combate y después de calculados y visualizados todos sus efectos, comienza la vez del segundo jugador.





Cortesía de Soft

Eastern Front (Frente del Este)
En este juego el jugador asume el rol del ejército alemán intentando alcanzar Moscú en 1941, mientras que el ordenador actúa defendiendo las fuerzas rusas. Escrito por Chris Crawford y distribuido por Atari, incorpora configuraciones nuevas, como *fine scrolling*. Una de las configuraciones más atractivas de *Eastern Front*, desde el punto de vista visual, es la forma en que se modifica el mapa a medida que transcurre el año. En otoño los árboles pierden las hojas y luego, en invierno, los ríos se hielan y el suelo se cubre de nieve. En este juego, llegar a Moscú se considera algo muy difícil

Ian McKinnell

Para muchas personas la fascinación que ejercen los juegos estratégicos nace del hecho de que no existe ninguna solución "correcta" a los problemas que plantea el juego. El placer del jugador deriva de la superación de los problemas físicos y logísticos del terreno en el que está operando, así como del reto intelectual que supone utilizar los recursos disponibles para derrotar al enemigo. Naturalmente, a todos los estrategas les gustaría ganar con los esquemas más atrevidos y las trampas más cuidadosamente urdidas; pero, sobre todo, ¡lo que les interesa es ganar!

Herramientas

Los equipos de herramientas son paquetes de software que mejoran una versión limitada de BASIC y ofrecen al programador utensilios para corregir errores

Los primeros ordenadores personales, como el Apple II y el Commodore PET, tenían capacidades limitadas y se diseñaron fundamentalmente para manipular números y texto. Del BASIC con que se dotaba a estas máquinas sólo se exigía que proporcionara órdenes y rutinas con estos fines. Como resultado de ello, se escribieron muchos programas de "utilidades" o de "equipos de herramientas", por lo general en código de lenguaje máquina, que operaban desde fuera del área de programación en BASIC. Éstos proporcionaban medios auxiliares para programación en forma de órdenes directas adicionales que podían ser de ayuda en la construcción y depuración de programas.

Desde entonces los ingenieros han creado una multitud de capacidades para gráficos y sonido, como consecuencia del creciente interés en los juegos recreativos para ordenadores personales. Cada nuevo modelo introduce configuraciones más amplias que enseguida se incorporan al software escrito profesionalmente. Sin embargo, salvo una o dos excepciones, el BASIC incorporado añade pocas mejoras, cuando no ninguna, respecto a las primeras versiones. Por ende, el usuario ha de elaborar rutinas, a menudo empleando órdenes PEEK y POKE repetidas, para incorporar estas nuevas configuraciones a la gama de órdenes disponibles. Consecuencia de todo esto es que ahora existen muchas utilidades, equipos de herramientas y ampliaciones para la mayoría de las máquinas populares. Se ha conseguido con ello un acceso más sencillo a las facilidades existentes (p. ej., a los editores de sprites o de sonido), una ampliación del software (p. ej., creadores de sprites) o simplemente ayudas para la programación en BASIC.

Las ampliaciones de esta clase se pueden localizar en RAM, en ROM interna o en cartucho de ROM. Es preferible una ampliación de ROM a una cargada en RAM, ya que no ocupa la memoria del usuario y está protegida contra un borrado por descuido. Por lo general, un programa escrito con la ayuda de un equipo de herramientas sólo funcionará en otro ordenador que esté similarmente equipado. No obstante, existen utilidades generadoras de programas autónomos, que se pueden ejecutar en una versión no ampliada del ordenador. Ésta es la base de la mayoría de los editores de gráficos y de sprites, así como de algunos editores de sonido.

Entre las configuraciones útiles que cabe hallar en las ampliaciones al BASIC están las órdenes especiales para gráficos (como PAINT, DRAW, PLOT, CIRCLE, etc.) y las órdenes para sonido (como SOUND, PLAY, MUSIC, ENVELOPE, etc., o palabras que describen un efecto sonoro, como BANG o ZAP). Otras facilidades útiles son las órdenes para programación estructurada, como REPEAT...UNTIL e IF... THEN...ELSE. Las sentencias de este tipo le permiten al usuario escribir programas que vayan avanzando en secuencia lógica, y evitan el código desaliñado y difícil de comprender que resulta de la utilización indiscriminada de GOTO.

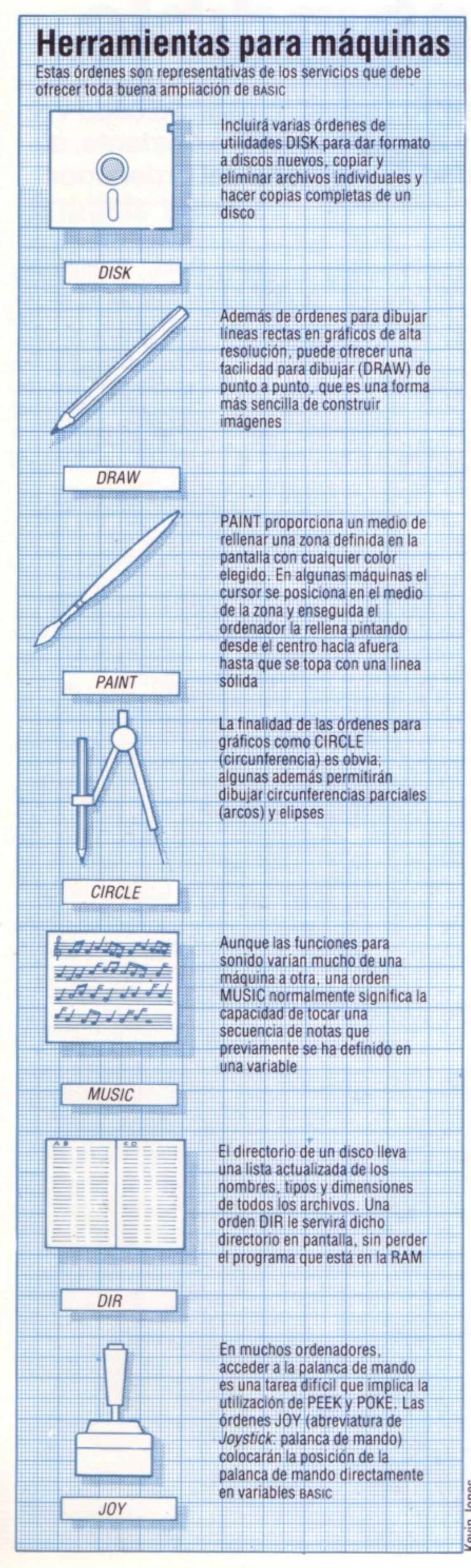
Cajas de herramientas
He aquí algunos de los equipos de herramientas y paquetes de ampliación del BASIC disponibles para algunos de los ordenadores personales más populares. Los paquetes que crean una facilidad de sprites en los ordenadores que carecen de esta configuración están adquiriendo una creciente popularidad

Equipos de herramientas

SUPER TOOL KIT	De Nectarine, disponible para los Spectrum de 16 K y 48 K
SPECTRUM EXTENDED BASIC	De CP Software, para el Spectrum de 48 K
SPECTRUM KEYDEFINE	De Scientific Software, para el Spectrum de 48 K
PROGRAMMER'S AID	De Commodore, para el Vic-20
BUTI	De Audiogenic, para el Vic-20
TOOL BOX	De BBC Software, para los BBC Modelos A y B
SPRITE MAGIC	De Merlin Micro Systems, disponible para el Dragon 32
SPRITE GRAPHICS	De B Sides Software, para el Spectrum de 48 K
SPRITE MASTER	De Micro Dealer UK, para el BBC Modelo B

El BASIC de Simon

Actualmente, la ampliación más completa para el lenguaje BASIC es el BASIC de Simon, disponible para el Commodore 64 en forma de un cartucho de ROM. El BASIC Commodore estándar, incorporado en el modelo 64, está bastante anticuado en el sentido de que apenas si proporciona órdenes especializadas y ninguna para la programación estructurada. Aunque sí posee algunas configuraciones de hardware avanzadas, como un amplio sintetizador de sonido, gráficos en alta resolución y gráficos sprite, el control del BASIC sobre estas funciones se efectúa a través de PEEK y POKE. El Simon le proporciona al Commodore una considerable ampliación, en virtud de las siguientes facilidades extras:



Kevin Jones

1) Un extenso acervo de recursos auxiliares para programación, incluyendo funciones que aseguran un control adicional sobre el listado y la depuración del programa y las ayudas de seguridad (protección del programa contra copias no autorizadas).

2) Órdenes adicionales para el manejo de variables y la manipulación de textos.

3) Operadores aritméticos extras y órdenes para conversión numérica.

4) Órdenes simplificadas para manipulación de discos.

5) Órdenes para gráficos en alta resolución que permiten mezclar texto con el trazado de puntos y dibujos. Se incluye también una facilidad para colorear croquis.

6) Órdenes para gráficos en baja resolución y manipulación de pantalla que pueden duplicar las áreas asignadas para gráficos y manipular con gran facilidad el área de pantalla. También permiten guardar en disco, cinta o impresora el contenido de cualquier pantalla.

7) Generador y editor de sprites fácil de usar.

8) Programación estructurada empleando órdenes para procedimientos como PROC, CALL y EXEC; más rutinas para bucles y comprobación de condiciones, como REPEAT...UNTIL, LOOP...EXIT, IF...END LOOP e IF...THEN...ELSE, que generalmente eliminan la necesidad de sentencias GOTO y GOSUB.

9) Rutinas para la creación de sonidos que permiten acceder a la gama completa de facilidades de sonido del 64 utilizando órdenes simples para dar forma a sonidos o tocarlos.

10) Órdenes simples para lápiz óptico, palancas de mando y mandos de raqueta.

Es muy poco frecuente que una ampliación incluya una gama de rutinas adicionales tan completa. La mayoría de los paquetes proporcionan utilidades y órdenes para un área específica de la programación. Por ejemplo, el cartucho Super Expander de Commodore, para el Vic-20, sólo suministra una gama simple de órdenes para música y gráficos en alta resolución. Las ampliaciones más populares son las ayudas para la construcción de programas. Éstas generalmente proporcionan órdenes de entrada de teclas únicas y diversas rutinas automáticas que simplifican la numeración de sentencias, la edición y la depuración en la modalidad directa.

Es un hecho normal que las utilidades y ampliaciones aporten más facilidad de acceso a las capacidades incorporadas del ordenador. Pero las rutinas que mejoran estas capacidades son más difíciles de encontrar, aunque están ya saliendo al mercado paquetes muy ingeniosos. Por ejemplo, las muchas ventajas que ofrecen los gráficos sprite para los juegos recreativos de acción rápida han sido la fuente de inspiración de algunas empresas más dinámicas para componer utilidades generadoras de sprites para aquellos ordenadores que no disponen de esta configuración.

Las utilidades, los equipos de herramientas y las ampliaciones para el BASIC que hemos reseñado aquí constituyen una pequeña fracción de las mejoras y las ayudas existentes. Aunque la tendencia actual de los fabricantes se orienta a proporcionar versiones de BASIC amplias y avanzadas, siempre existirá la necesidad de ayudas de software para contribuir a que la programación no sea una pesada carga sino un placer creativo.

La voz inconfundible

Los sistemas de reconocimiento de voz se están usando cada vez más en aplicaciones comerciales y de seguridad. No obstante, su poder se ve limitado por la capacidad de memoria del ordenador

Para que un ordenador sirva para algo ha de disponer de algún medio viable que permita alimentar en él las órdenes y la información. La interface que normalmente empleamos para comunicarnos con un ordenador personal es un teclado (aunque los "ratones" y las palancas de mando constituyen posibles alternativas). No obstante, utilizando un teclado nos encontramos con que nos vemos obligados a comunicarnos con el sistema mediante un lenguaje artificial. Las órdenes como CLS, DIRECTORY, RUN, LOAD y SAVE pueden tener algún significado para el sistema operativo, pero no son "naturales".

El sistema de comunicación natural entre los humanos es el habla, y no digitar los mensajes en teclados y contemplar las respuestas en aparatos de televisión. Si se pudiera lograr que un ordenador comprendiera las órdenes orales (aunque estuvieran expresadas en la misma forma que las consignadas a través de un teclado), éste sería mucho más fácil de utilizar, especialmente por aquellas personas con alguna incapacidad física. Para que un sistema de ordenador pueda "comprender" palabras habladas, primero debe procesar la entrada de sonido: las señales analógicas se deben analizar y convertir en una forma digital que pueda manejar el ordenador. Aunque parece ser algo muy sencillo de generar electrónicamente, la voz es una combinación de sonidos extraordinariamente compleja.

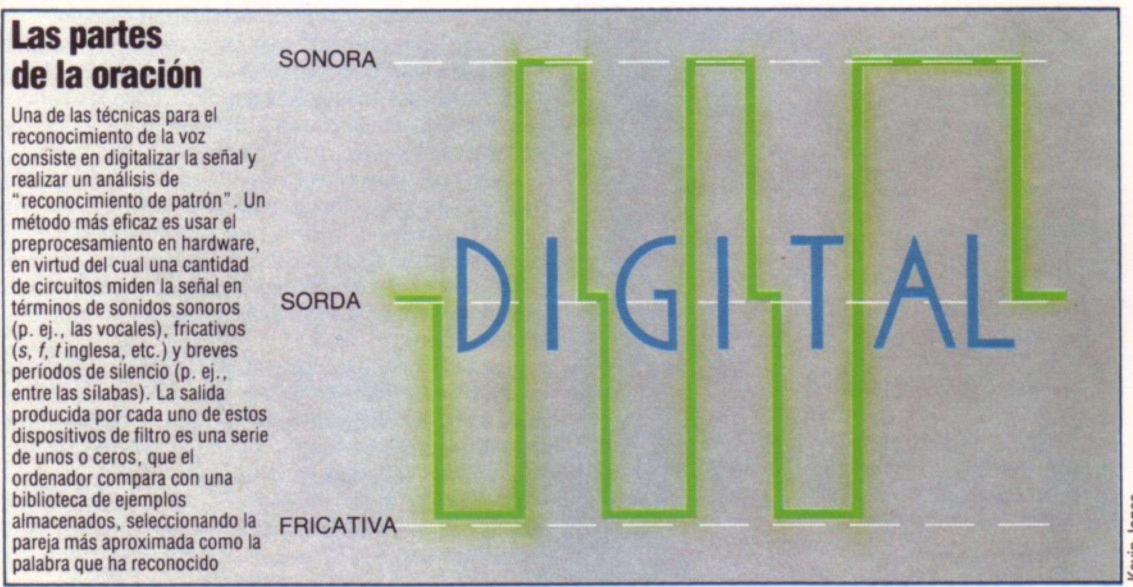
Los sueños del reconocimiento completo e instantáneo de voz (tal como tipifica el ordenador HAL en *2001: una odisea del espacio*) es muy poco probable que lleguen a ser realidad durante muchos años, si es que se llegan a conseguir. La máquina de escribir para entrada de voz está igualmente lejana; y, a pesar de todo, la tecnología tanto para esta máquina como para el ordenador "comprensivo" ya existe. Pero ninguna de las dos se puede conseguir por un precio reducido, porque existe una enorme dificultad en cuanto a crear sistemas de reconocimiento de voz: las palabras pueden sonar iguales pero tener significados diferentes, según el contexto en el que aparezcan. El potencial de procesamiento necesario para solventar este problema simplemente no se logra a un precio razonable.

A pesar de que los investigadores han creado sistemas que se aproximan a este objetivo, han descubierto que al aumentar el número de hablantes que el ordenador puede reconocer se reduce el número de palabras que se pueden reconocer por vez. Por lo general, un sistema de reconocimiento de hablantes múltiples permitirá reconocer entre 20 y 30 palabras al mismo tiempo, con un margen de acierto de un 85 o 90 % aproximadamente.

Las aplicaciones potenciales de los sistemas de reconocimiento de voz son considerables. La Oficina de Correos alemana utiliza uno de ellos para ayudar a clasificar la correspondencia; y en la actualidad existen muchas aplicaciones en el ámbito aeroespacial, tanto militar como civil, donde los pilotos, pese a utilizar manos y pies, se ven desbordados en su esfuerzo por controlar los aparatos. En todas estas situaciones el número de palabras que se pueden reconocer cada vez se limita a alrededor de 20. Sin embargo, esto no significa que el sistema global sea limitado. El usuario selecciona una palabra de entre las 20 de un menú, y cada orden reconocida produce un nuevo menú de palabras entre las que escoger. El ordenador sólo emprenderá alguna acción después de que consiga reconocer la

Las partes de la oración

Una de las técnicas para el reconocimiento de la voz consiste en digitalizar la señal y realizar un análisis de "reconocimiento de patrón". Un método más eficaz es usar el preprocesamiento en hardware, en virtud del cual una cantidad de circuitos miden la señal en términos de sonidos sonoros (p. ej., las vocales), fricativos (*s, f, t* inglesa, etc.) y breves períodos de silencio (p. ej., entre las sílabas). La salida producida por cada uno de estos dispositivos de filtro es una serie de unos o ceros, que el ordenador compara con una biblioteca de ejemplos almacenados, seleccionando la pareja más aproximada como la palabra que ha reconocido



secuencia completa. En el caso de la oficina de clasificación, el primer nivel de clasificación sería por país y, una vez seleccionado el país correcto, la siguiente clasificación sería por provincia, luego pueblo, etc. El envío sólo se remitiría hacia su destino en el nivel inferior, asegurando de esta manera la máxima fiabilidad de la operación.

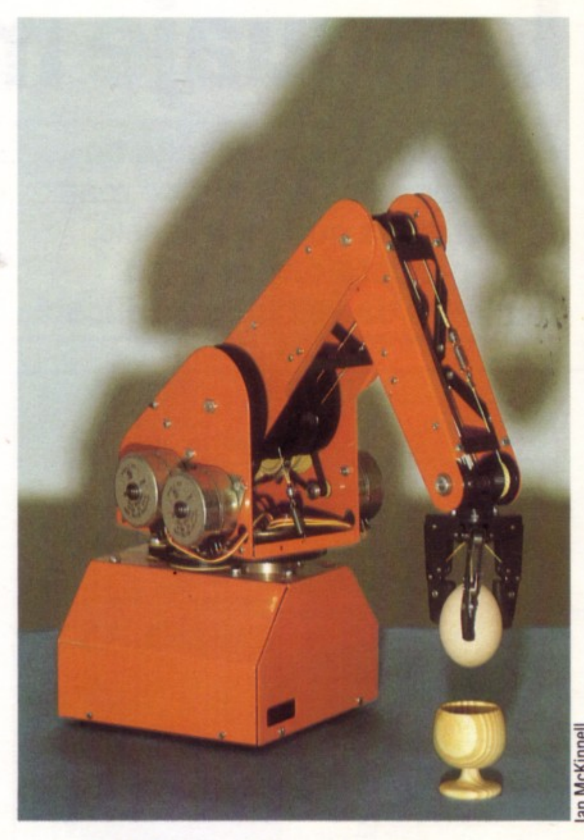
Análisis de voz

El reconocimiento de voz se suele abordar de dos maneras. El método "rápido" consiste simplemente en alimentar la voz a través de un convertidor analógico-digital, y utilizar el potencial del ordenador para llevar a cabo todo el análisis. Lamentablemente, este procedimiento posee algunos inconvenientes, siendo el principal el tiempo que ocupa realizar el análisis. Los sistemas que emplean este método pueden tardar dos o tres segundos en reconocer la entrada. Para que el reconocimiento de voz sea verdaderamente útil, el ordenador debe "comprender" el habla con la misma rapidez que un ser humano, y el enfoque de procesamiento de números raramente lo consigue.

El otro método consiste en emplear el preprocesamiento. En vez de analizar la señal de voz matemáticamente, se puede hacer gran parte del trabajo mediante la electrónica estándar. Lo que se le proporciona entonces al ordenador es información acerca de la entrada hablada: de la frecuencia, contenido, tono, energía, etc. Las frecuencias se pueden medir filtrando la señal y detectando el nivel en cada banda de frecuencia, algo así como utilizar los controles de tono en un equipo de alta fidelidad para "producir" el bombo. Dado que todo este procesamiento electrónico se realiza al mismo tiempo que se alimenta a los circuitos la señal de habla original, el análisis es casi instantáneo. Efectuar una operación similar en los datos de un convertidor analógico-digital requeriría que varios ordenadores trabajaran con los números al mismo tiempo. El método de preprocesamiento está todavía en fase de investigación pero parece ser el que presenta mayores posibilidades de éxito.

Una vez que se ha extraído de la señal original (independientemente del método empleado) la información relativa a frecuencia de contenido, tono, energía, etc., se lleva a cabo el verdadero reconocimiento, comparando el conjunto de las cifras obtenidas con una cantidad de modelos almacenados en la memoria del ordenador. Estos modelos se crean "entrenando" al sistema de reconocimiento. Las palabras que se van a reconocer se le dictan de viva voz al sistema de a una por vez y la información resultante se almacena en una "biblioteca" digital de ejemplos. Entonces se vuelve a dictar el conjunto completo de palabras y el ordenador compara la entrada con el modelo que tiene de ella. Si coinciden, al primer juego de información se agrega otro, para formar una versión más completa del modelo.

Para reconocer una palabra hablada, el ordenador debe emparejar el patrón de información de la entrada con uno o varios de los modelos almacenados en la biblioteca habitual. En muchos casos se hallarán varias posibles parejas que forman parte de otras palabras que concuerdan con el patrón de entrada. Las dos primeras sílabas de "internacional", por ejemplo, son las mismas que las de "inter-


Ian McKinnell

Control del entorno
Las aplicaciones del reconocimiento de voz más recientes son de naturaleza educativa. Una de éstas es la denominada *entorno limitado*, en la que intervienen un ordenador, un brazo-robot y un número de objetos sencillos que el brazo pueda manipular. Hablando por un micrófono, el usuario puede indicarle al brazo que «COLOQUE EL HUEVO EN LA HUEVERA». El ordenador tendrá que interpretar las órdenes y buscar en su memoria las posiciones de los objetos

ventor". Al final de la búsqueda, debería haber una palabra que resaltara como la pareja más perfecta en relación a todas las otras posibilidades, y es ésta la que el ordenador interpretará como la entrada.

Con toda seguridad, las facilidades para reconocimiento de voz hallarán en el futuro muchas aplicaciones, pero es probable que se utilicen más directamente en paquetes complejos de software, como bases de datos, donde las órdenes se seleccionan de un menú que aparece en la pantalla. Este tipo de aplicación eliminará el mayor obstáculo con el que se encuentran quienes no son expertos: el teclado. Los sistemas de videotex han reducido el dispositivo de entrada a un sencillo teclado numérico adicional, pero éste limita sustancialmente el grado de interacción que puede conseguir el usuario. Una interface activada mediante la voz que pudiera reconocer un juego estándar de órdenes de interrogación para bases de datos, así como los símbolos numéricos y las letras del alfabeto, proporcionaría una eficaz configuración.

Existen en la actualidad unidades de reconocimiento disponibles a nivel comercial que se pueden enchufar en ordenadores personales, pero se trata de dispositivos muy poco sofisticados. Sistemas como Big Ears y Speech Lab, de Heuristic Inc., emplean gran potencial de procesamiento para reconocer apenas unas pocas palabras pronunciadas por una persona. Lo que se necesita para que el reconocimiento de voz sea verdaderamente útil es una capacidad para reconocer las palabras pronunciadas por *cualquier* persona, independientemente del dialecto o el acento. En esta etapa, el factor restrictivo es la cantidad de memoria disponible para retener los modelos. Una posibilidad interesante consiste en utilizar un videodisco para retener un juego estándar de modelos: éste apenas si emplearía memoria interna y prácticamente no se notaría ninguna reducción de velocidad.

Lenguaje máquina

El aprendizaje del código de lenguaje máquina ofrece más dificultades que el del BASIC, pero proporciona un gran aumento de velocidad y eficacia

Paso a paso

El programa en código de lenguaje máquina se almacena en una zona de la memoria, si bien los datos sobre los que operará pueden estar en cualquier otro lugar de la memoria. Observe que los operandos (p. ej., $3F80) se almacenan como dos bytes, con el byte inferior ($80) antes del byte superior ($3F)

Programa
LDA
80
3F
ADC
81
3F
STA
93
04

El contador del programa (PC) es un registro en el interior de la CPU que indica la instrucción que se está ejecutando

CPU

Dirección de memoria

$ 0493 08 STA $0493

$3F80 03 LDA $3F80

$3F81 05 ADC $3F81

Todos los programas en código de lenguaje máquina se construyen a partir de operaciones simples que transfieren bytes de memoria a los registros internos de la CPU, los procesan y los devuelven a una posición de la memoria. Este diagrama muestra el programa que se requiere para sumar los contenidos de dos posiciones y almacenar el resultado en una tercera

La primera instrucción carga en el acumulador el contenido de la posición $3F80 (es decir, el valor 03). La segunda le agrega al acumulador el contenido de $3F81 (es decir, el valor 05). La tercera almacena el nuevo contenido del acumulador (ahora 08) en la posición de memoria $0493

Kevin Jones

Hasta este momento en *Mi Computer* la programación se ha centrado en el lenguaje BASIC, porque es a la vez versátil y fácil de utilizar. Sin embargo, a medida que su experiencia aumente y que los proyectos de programación que emprenda se vayan haciendo más ambiciosos, no tardará en descubrir las limitaciones de este lenguaje.

Por el contrario, la programación en código de lenguaje máquina supone muy pocas restricciones en cuanto a lo que usted puede hacer y, en comparación con el BASIC, da la sensación de una velocidad casi infinita. Así y todo, muy pocos usuarios de ordenadores personales se atreven a dar el salto del BASIC al lenguaje máquina, en parte porque la utilización del código de lenguaje máquina es un proceso de programación que exige un trabajo mucho más minucioso, y también porque conceptualmente se diferencia mucho del BASIC y de todos los otros lenguajes de alto nivel. A pesar de ello, vale la pena tener idea del código de lenguaje máquina; y en este capítulo, al que seguirá otro más, analizaremos los procedimientos fundamentales.

Como ya explicamos, el lenguaje máquina es el único que comprende el microprocesador (la CPU), auténtico cerebro del ordenador. Este microprocesador sólo sabe realizar funciones muy sencillas (p. ej., puede sumar dos dígitos de un número, pero no sabe multiplicarlos). Sin embargo, realiza estas funciones a velocidades asombrosas. Cada operación de un microprocesador se caracteriza por el número de *ciclos de reloj* que necesita. Si la CPU de su ordenador funciona a 1 MHz, entonces un ciclo de reloj equivale a un microsegundo, y una operación cuya ejecución requiera cuatro "ciclos de reloj" se realizará en cuatro millonésimas de segundo.

Por consiguiente, todo programa escrito en lenguaje máquina constará de un gran número de instrucciones y todas las funciones se han de construir "a mano" a partir de operaciones simples. Toda programación en código de lenguaje máquina consiste en la manipulación de bits y bytes individuales de la memoria, utilizando funciones lógicas simples como AND, OR y NOT, y una elemental aritmética en sistema binario.

Ésta es una de las razones por las cuales escribir en código de lenguaje máquina es una tarea tan lenta; la otra es que el programador tiene que saber cuál es el lugar de la memoria en que se conserva todo. En BASIC, cada vez que se encuentra una sentencia como LET A = 5, es tarea del intérprete encontrar un espacio de la memoria donde almacenar esa variable. Y cada vez que más adelante en el programa se haga referencia a A, recordará dónde buscar los datos necesarios. Cuando se comienza a programar en código de lenguaje máquina, uno descubre que debe especificar una dirección (una posición de memoria) para cada dato que necesite almacenar, y que ha de asegurar que encima no se escriban por descuido otros datos diferentes.

Analicemos en qué consiste el lenguaje máquina. (A propósito, todos nuestros ejemplos se refieren a unidades CPU de ocho bits, como el Z80 y el 6502; los dispositivos de 16 bits trabajan de modo similar pero a cada operación procesan el doble de bits.) El microprocesador está conectado a la memoria del ordenador mediante dos buses (un bus no es más que un grupo de cables o líneas): el bus de direcciones y el bus de datos (véase p. 144). También existe el llamado bus de control, pero éste sólo proporciona a la CPU señales de sincronización y el programador no lo utiliza.

El bus de direcciones tiene una anchura de 16 bits y, colocando en este bus un patrón de bits, la CPU puede seleccionar cualquiera de los 65 536 bytes de su "mapa de memoria" (véase p. 329). En un ordenador personal típico, algunas de estas posiciones de memoria son RAM, otras ROM, otras chips especiales de entrada-salida y hay algunas más sin emplear. Si la CPU desea leer una posición de memoria (una de las líneas del bus de control indica si se ha de efectuar una lectura o una escritura), entonces el byte seleccionado colocará su contenido en el bus de datos, en forma de un patrón de ocho bits. Del mismo modo, la CPU puede escribir un patrón de ocho bits en cualquier posición selec-

cionada. La CPU no sabe qué partes de la memoria son ROM y RAM, y dar con las direcciones correctas es otro de los aspectos cruciales que se dejan al programador.

Dentro del microprocesador existen quizá media docena de "registros", que son como posiciones individuales de memoria y que se utilizan para almacenar resultados temporales y realizar las funciones de lógica y de aritmética binaria. La mayoría de estos registros equivalen a un byte de memoria, aunque algunos son de 16 bits de anchura. Uno de estos últimos se denomina registro *contador de programa* (PC) y éste contiene la dirección en la memoria de la instrucción en código de lenguaje máquina que se está realizando en esos momentos. Podemos imaginárnoslo como la línea en un programa escrito en BASIC.

Otro registro importante (pero en esta ocasión de sólo ocho bits de anchura) es el *acumulador*. Como su nombre indica, este registro puede acumular totales (es decir, se le pueden sumar o restar bytes) y en general es el único registro que puede efectuar toda clase de operaciones aritméticas. Por tanto, un programa muy sencillo en código de lenguaje máquina se podría especificar así:

1) Cargar en el acumulador el contenido de la posición de memoria $3F80. En código de lenguaje máquina las direcciones se suelen escribir en hexadecimal (véase p. 179). Los números hexadecimales se indican anteponiéndoles un signo especial, por lo general $.

2) Sumarle al acumulador el contenido de la posición de memoria $3F81, dejando abierta la posibilidad de que el resultado pueda ser mayor de lo que se puede almacenar en un solo byte, en cuyo caso también habrá un "bit de arrastre" (denominado en inglés *carry bit*).

3) Almacenar el nuevo contenido del acumulador (es decir, el resultado) en la posición de memoria $0493.

Cada uno de estos puntos constituye una instrucción en código de lenguaje máquina, y normalmente el programa se escribirá de la manera siguiente:

```
LDA $3F80   (LoaD Accumulator) (cargar acumulador)
ADC $3F81   (ADd with Carry) (sumar con posible arrastre)
STA $0493   (STore Accumulator) (almacenar acumulador)
```

Las observaciones entre paréntesis, al igual que las sentencias REM en BASIC, no poseen ningún efecto. La primera entrada de cada línea se denomina *opcode* (código de la operación) e indica la naturaleza de la operación a realizar. La segunda columna contiene el "operando": los detalles, o el paradero, del dato sobre el cual se ha de operar. Un microprocesador normalmente admitirá varias docenas de posibles *opcodes* (es decir, puede realizar varias docenas de tipos de operaciones simples) y, cuando se le haya dado entrada en la máquina, cada *opcode* ocupará tan sólo un byte de la memoria.

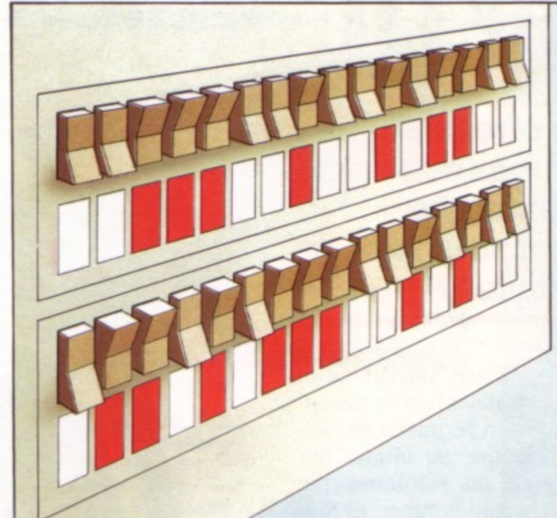
Por consiguiente, un *opcode* se puede especificar como un número dentro de la escala 0-255 (o, mejor dicho, dentro de la escala hexadecimal entre $00 y $FF). Sin embargo, mientras se está desarrollando un programa, para que el listado resulte más legible se utilizan tres letras mnemotécnicas, como LDA, ADC y STA.

Cada uno de los tres operandos mencionados se compone de un número hexadecimal comprendido en la escala $0000-$FFFF, y emplean hasta dos bytes del espacio de memoria para el programa. No obstante, algunos operandos sólo tienen un byte de longitud, y algunos *opcodes* no poseen ningún operando. El breve programa que hemos visto ocuparía, por tanto, un total de sólo nueve bytes, sin incluir las tres posiciones de memoria ($3F80, $3F81 y $0493) sobre las que operará el programa. Para este ejercicio trivial, el siguiente programa en BASIC alcanzaría exactamente los mismos objetivos, pero ocuparía alrededor de 50 bytes y efectuaría la operación cien veces más lentamente, en virtud de todo el tiempo que invertiría el intérprete en traducirlo:

```
10 A = PEEK (16256)
20 A = A + PEEK (16257)
30 POKE 1171,A
```

Nota: Quizá las posiciones utilizadas para este programa no sean adecuadas para su máquina.

En el próximo artículo analizaremos cómo se da entrada al código de lenguaje máquina en un ordenador personal y cómo se ejecuta.



Luces intermitentes
La idea de los inmensos paneles de luces que suelen caracterizar los ordenadores en las películas proviene del "panel frontal" que poseen muchos miniordenadores. Este panel frontal era una línea de luces e interruptores que representaban los buses de direcciones y de datos de la CPU. Antes de que los teclados se conectaran mediante interfaces, se había de dar entrada a todos los programas en código de lenguaje máquina de esta forma, en sistema binario

Kevin Jones



LDA (LoaD Accumulator)
Transfiere los contenidos de una única posición de memoria (byte) al registro del acumulador interno

STA (Store Accumulator)
Realiza el proceso opuesto que LDA

ADC (ADd with Carry)
Suma el contenido de una posición de memoria al contenido actual del acumulador, creando, de ser necesario, un bit "para llevar"

SBC (SuBtract with Carry)
Ésta es la función inversa a ADC

JMP (JuMP)
Transfiere la operación del programa a una nueva posición. Es similar en operación a una sentencia GOTO en BASIC

Opcodes
He aquí algunos de los opcodes (códigos de operación o instrucciones) que puede ejecutar un microprocesador normal

Sinclair QL

El Quantum Leap ofrece el microprocesador más avanzado de todos los ordenadores personales, con un potencial para medio megabyte

Todas las innovaciones de sir Clive Sinclair en el campo de los ordenadores personales han representado saltos cuantitativos tanto en términos de tecnología como de precio, sólo que su último microordenador es la primera de sus máquinas que declara esta verdad ya en su propio nombre: el Sinclair Quantum Leap (QL), que literalmente significa "salto cuantitativo". Por su precio bastante moderado, está pensado para un número creciente de usuarios que son o bien grandes entusiastas de los ordenadores o que pretenden alguna aplicación tanto de gestión como personal. Como tal, representa una competencia muy seria para máquinas como el Commodore 64 y el BBC Modelo B, aunque por su especificación técnica es a todas luces superior.

Resulta bastante evidente que el QL nació como un resumen de todos los componentes y configuraciones más novedosas en cuanto a ordenadores. Abriendo una brecha en la opción normal entre el Z80 o el 6502, la CPU es un miembro de la familia Motorola 68000, que actualmente es el microprocesador más exquisito que puede hallarse en un microordenador, y que utilizan máquinas como el Lisa de Apple (véase p. 261). No obstante, la CPU es un 68008, lo que significa que si bien sus registros internos son de 16 bits (y puede realizar muchas funciones a través de 32 bits completos), su bus de datos externo sólo tiene una anchura de ocho bits. Esto retardará la operación de la CPU muy ligeramente, porque la carga y el almacenamiento de los registros se habrá de hacer por mitades. Pero esto también significa que se mantiene reducido el costo de los chips de memoria y, a la hora de elegir los componentes, la economía suele ser una de las consideraciones básicas de Sinclair.

El QL viene con 128 Kbytes de RAM como estándar, pero con futuros accesorios se podrán ampliar a 512 Kbytes ("medio mega", como se suele decir). Esta vasta memoria es particularmente útil para aplicaciones de gestión empresarial, ya que reduce la frecuencia con la cual el programa se debe remitir al almacenamiento fuera de línea. Este al-

Interface para ampliación
Capaz para acoplar periféricos y hasta 0,5 Megabytes de RAM

Ranura para cartucho ROM
Aquí se pueden enchufar hasta 32 K de ROM adicional

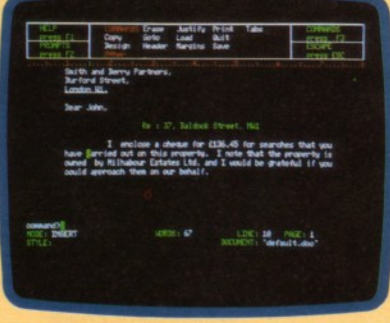
Conexiones para palanca de mando

Microprocesador 68008
Este procesador configura registros internos de 16 y 32 bits, con un bus de datos externo de 8 bits

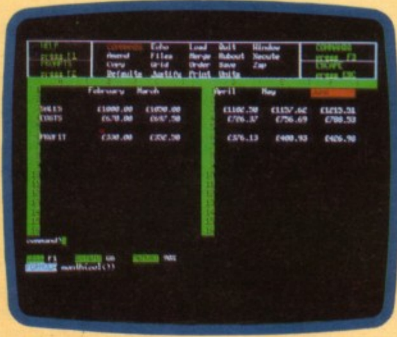
Chips a la medida
Es cada vez mayor el número de ordenadores nuevos que configuran un chip diseñado a medida. El QL posee dos, para manipular la visualización y diversas interfaces

Ian McKinnell

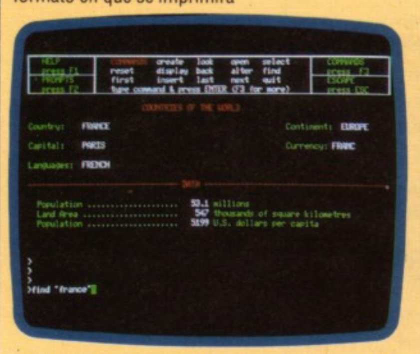
El software del QL



QL Quill es un paquete para tratamiento de textos que visualiza el texto en la pantalla en el mismo formato en que se imprimirá



QL Abacus es una hoja electrónica en la que se pueden señalar las celdas por su nombre en lugar de mediante las coordenadas



QL Archive es un paquete para base de datos. El usuario puede diseñar trazados para informes, valiéndose de la ayuda de un editor de pantalla.



QL Easel, utilidad para gráficos y diagramas, maneja aspectos del diseño como la reducción y ampliación a escala



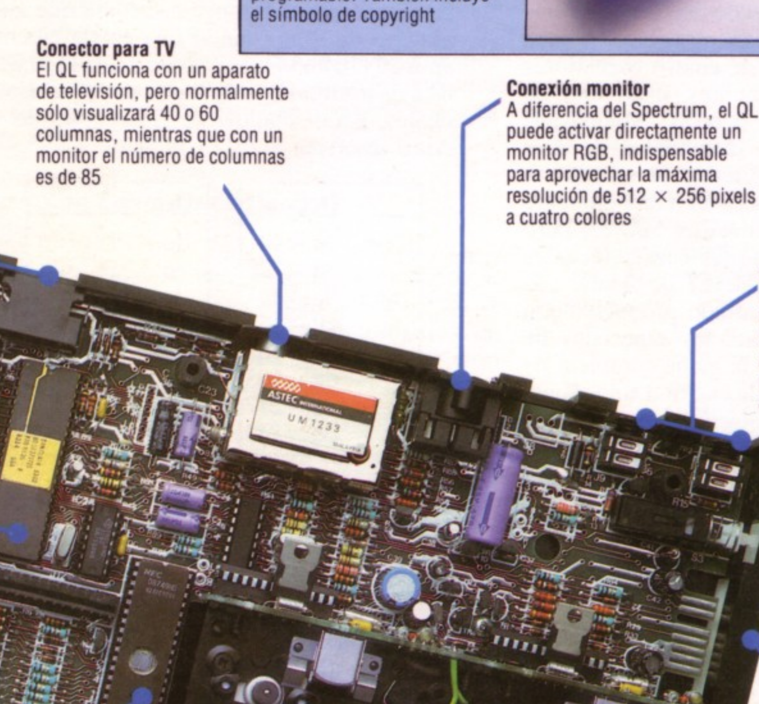
Teclado
El teclado se basa en una construcción de tipo membrana (que lo protege contra accidentes como derramamiento de café, salpicaduras, etc.), pero contiene 65 teclas de recorrido total y la "sensación táctil" es tan buena como la de algunas de las máquinas de gestión más caras. Hay cuatro teclas para control del cursor y cinco teclas de función programable. También incluye el símbolo de copyright

Puertas en serie
Incorpora dos puertas RS232, aptas para activar una impresora y un modem. La interface para impresora más común (Centronics) se debe adquirir como accesorio

Conector para TV
El QL funciona con un aparato de televisión, pero normalmente sólo visualizará 40 o 60 columnas, mientras que con un monitor el número de columnas es de 85

Conexión monitor
A diferencia del Spectrum, el QL puede activar directamente un monitor RGB, indispensable para aprovechar la máxima resolución de 512 × 256 pixels a cuatro colores

Interface para red
Se pueden conectar entre sí hasta 64 ordenadores QL y Spectrum (estos últimos agregando la interface 1) para conformar una red de área local



Ian McKinnell

Ranura para ampliación de microdrive
Al igual que el Spectrum, el QL puede manejar hasta ocho microdrives

Microdrives
Cada uno de éstos utiliza un diminuto cartucho de wafer, que contiene un bucle continuo de cinta para almacenar hasta 100 K cada uno

Segundo microprocesador
Este Intel 8049 controla el teclado, el sonido y las puertas en serie, dejando libre al 68008 para la ejecución de los programas del usuario

PROTOTIPO DEL TABLERO
En la fotografía vemos el tablero de circuito impreso de un QL prefabricado. Algunos extremos pueden variar en los modelos fabricados

SINCLAIR QL

DIMENSIONES
472 × 138 × 46 mm

CPU
Motorola 68008

VELOCIDAD DEL RELOJ
7,5 MHz

MEMORIA
128 K de RAM, ampliables a 512 K
32 K de ROM, ampliables a 64 K

VISUALIZACION EN VIDEO
25 líneas de 85 caracteres (con monitor), gráficos de alta resolución: 512 × 256 pixels (4 colores), 256 × 256 (8 colores)

INTERFACES
RS232 en serie (2), palancas de mando (2), microdrives, LAN, TV, monitor RGB

LENGUAJE SUMINISTRADO
BASIC

OTROS LENGUAJES DISPONIBLES
Existen planes para varios, entre los que destaca el lenguaje "C"

VIENE CON
Manual de instrucciones, cuatro programas de aplicaciones

DOCUMENTACION
El manual provisional es de un nivel elevado, viene en carpeta de anillas e incluye manuales para el software estándar

macenamiento se compone de dos microdrives incorporados en la carcasa, que ofrecen alrededor de 100 Kbytes cada una. Aunque esto hace que el QL sea un sistema de gestión autocontenido, los microdrives se deben considerar más bien como un punto débil en comparación con el procesador, que es notablemente eficaz. Localizar un dato en el microdrive cuesta unos 3,5 segundos, en contraste con el medio segundo que puede costar en la nueva generación de miniunidades de disco flexible.

Sinclair ha prometido producir una interface para una unidad de disco rígido (Winchester), pero no existe ningún plan para discos flexibles, aunque no cabe ninguna duda de que algunos fabricantes independientes los ofrecerán. Será una pena, pues sin disco el QL no podrá ejecutar el sistema operativo Unix, que se suele considerar como una de las principales razones para optar por una CPU Motorola 68000, llamada a suplir al CP/M como sistema operativo estándar para el software de gestión.

El QL viene con cuatro paquetes de gestión tipo, todos ellos elaborados por la empresa de software Psion. *Quill* es un procesador de textos; *Abacus*, un paquete de hoja electrónica; *Archive*, una base de datos, y *Easel*, un paquete para gráficos. Todos funcionan bajo el sistema operativo residente, que Sinclair ha apodado QDOS. La popularidad que probablemente conseguirá esta máquina implica que se desarrollará muchísimo software para ella, aunque a las firmas de software no les será fácil transferir al QL los paquetes ya existentes. Así y todo, se podría aducir que si Sinclair adoptara los estándares de la industria, sus productos no ocuparían el primer lugar en el mercado, como ahora.

El BASIC residente se ha mejorado respecto a la versión del Spectrum y, como si el propio nombre Quantum Leap no fuera ya de por sí suficientemente descriptivo, Sinclair lo ha denominado SuperBASIC. Incluye facilidades para manipular procedimientos (favoreciendo, por tanto, la programación estructurada) y para acceder al sistema operativo desde un programa en BASIC. Tanto éste como el QDOS están contenidos en los 32 Kbytes de la ROM estándar.

El Sinclair QL es indudablemente una máquina impresionante y, tal vez lo más importante de todo, posee las suficientes posibilidades de ampliación como para protegerse contra la obsolescencia.

Principios sonoros

Las funciones para sonido de los modelos Atari disponen de cuatro voces independientes

Las facilidades de sonido del Atari son buenas (tal como se desprende de muchos de los juegos en cartucho), aunque los medios para su control no son tan comprensibles. Dispone de cuatro osciladores de onda cuadrada independientes, cada uno de ellos con una escala de tres octavas. Además, la salida del oscilador se puede distorsionar de siete maneras para dar mayor variedad al sonido. A estas facilidades se puede acceder cómodamente desde el BASIC a través de la orden SOUND, pero ésta no hace un uso cabal de las configuraciones extras del chip para sonido POKEY de Atari, que puede modificar aún más el sonido producido con filtros "paso-altos" y modalidades especiales de operación. Por consiguiente, la gama completa de control de sonido sólo se puede explotar totalmente utilizando complicadas órdenes POKE o código de lenguaje máquina, que cae fuera de este nivel del curso. La salida se efectúa solamente a través del altavoz del televisor.

SOUND

Ésta es una orden muy sencilla, con el siguiente formato:

```
SOUND O,P,D,V
```

O = oscilador (0-3)
P = tono (0-255)
D = distorsión (1-15)
V = volumen (1-15)

Cada orden SOUND puede seleccionar sólo un oscilador, de modo que es imposible comenzar más de un oscilador a la vez. Esto no es un gran problema, pero si se programa música utilizando todos los osciladores para armonías de cuatro partes, la demora es perceptible.

El tono se calcula de forma un tanto extraña y, en consecuencia, algunas frecuencias son inexactas. La frecuencia disminuye a medida que aumenta el número de tono, dando una escala efectiva desde *do* a 29 (1046,5 Hz) hasta *do* a 243 (130,81 Hz). La lista da los números de tono para algunas notas musicales. En el manual de referencias del BASIC de Atari se ofrece la lista completa.

Octava-1
(Centr.) do – 121
si – 128
la – 144
sol – 162
fa – 182
mi – 193
re – 217
do – 243

Octava-3
do – 29
si – 31
la – 35
sol – 40
fa – 45
mi – 47
re – 53
do – 60

El parámetro de distorsión "P" equivale al canal de ruido que poseen la mayoría de los ordenadores, pero es muchísimo más versátil. Cada número par hace que se mezcle con la entrada estándar del osci-

Redondeos

Un somero análisis de los gráficos del Oric revela su semejanza con el Spectrum

El ordenador personal Oric-1 salió al mercado a mediados de 1983 y es obvio que se diseñó para que compitiera con el ZX Spectrum de Sinclair. El Oric ofrece cuatro modalidades de visualización. Sin embargo, sólo una modalidad permite la utilización de gráficos de alta resolución. Hay ocho colores disponibles; los colores de fondo y primer plano se establecen respectivamente mediante las órdenes INK y PAPER. El BASIC del Oric posee varias órdenes especiales de alta resolución para ayudar al programador de gráficos.

La pantalla se compone de 28 líneas, cada una de las cuales contiene 40 espacios de caracteres. Los caracteres del Oric no se diseñan utilizando la cuadrícula normal de ocho pixels por ocho, sino que se construyen sobre una de ocho por seis. En la modalidad de alta resolución, la resolución de la pantalla es de 240 × 200 pixels, y se reservan las tres líneas inferiores para información como mensajes de error, etc. No existe ninguna orden similar a PAINT, pero con poco esfuerzo se puede conseguir dicha función utilizando la orden FILL. Al igual que en el Spectrum (véase p. 392), se pueden mezclar gráficos de alta resolución con texto en la misma pantalla, pero el Oric permite colorear individualmente cada una de las líneas del cuadrado de un carácter, mientras que el Spectrum únicamente admite un color dentro del cuadrado de un carácter determinado.

Analicemos ahora con mayor detalle las modalidades en baja resolución que ofrece el Oric-1. El Oric posee tres modalidades en baja resolución: TEXT, LORES0 y LORES1. La única diferencia entre LORES0 y LORES1 es que utilizan juegos de caracteres diferentes. En la modalidad TEXT, las letras se

lador un arreglo diferente de impulsos al azar. Nótese que 10 da una señal libre de distorsión, y no 0, como cabría esperar. Un experimentado uso de sonidos distorsionados puede proporcionar timbres interesantes, y es particularmente útil para efectos especiales.

El volumen "V" se puede establecer entre 1 y 15, y un nivel intermedio razonable sería 7 u 8. Observe que no hay un modo adecuado de sincronizar la duración de las notas o los silencios entre ellas. En estas circunstancias el método normal consiste en utilizar bucles FOR...TO...NEXT sincronizados con sumo cuidado.

Para ilustrar el empleo de SOUND, las siguientes órdenes darán un *sol* no distorsionado en la octava 3 por el oscilador 1 a un volumen 8 para 50 pasos de bucles FOR...TO...NEXT:

```
10 SOUND 1,40,10,8
20 FOR N = 1TO50:NEXT N
30 END
```

El END de la línea 30 apaga todos los osciladores. Por otra parte, una nueva orden SOUND para el mismo oscilador interrumpe la nota antigua y da inmediatamente la nota nueva. He aquí un programa para tocar una melodía sencilla:

```
10 REM *DIXIE*
20 FOR I = 1TO7
30 READ N:REM *NOTA*
40 SOUND 3,N,10,7:REM *TOCAR NOTA*
50 FOR P = 1TO400:NEXT P:REM *PAUSA*
60 NEXT I
70 DATA 217,162,128,144:REM *RE SOL SI LA*
80 DATA 162,193,162:REM *SOL MI SOL*
90 END
```

A las capacidades de sonido del chip POKEY del Atari podemos llegar en BASIC colocando (POKE) números en las posiciones de memoria desde la 53760 a la 53763. Con este método, las rutinas de sonido se ejecutan con mayor rapidez y se pueden iniciar todos los osciladores a una. La información necesaria para conseguir esto, junto con algunas ampliaciones de conocimientos, incluyendo técnicas más atrevidas en código de lenguaje máquina, las encontrará en *De Re Atari*, que ofrece el Atari Program Exchange (APX), así como en el excelente *Atari Sound and Graphics*, publicado por John Wiley & Son.

pueden posicionar horizontalmente mediante la orden TAB. Sin embargo, en las otras dos modalidades LORES esta posibilidad se mejora pues permite especificar las posiciones verticales y las horizontales, utilizando la orden PLOTx,y,A$ donde *x* e *y* son las coordenadas de una posición de carácter determinada y A$ es la palabra o la frase a imprimir. El breve programa que sigue ilustra cómo se usan para escribir un nombre verticalmente:

```
10 REM LETRAS EN VERTICAL
20 CLS
30 LORES0
40 A$ = "RAMON"
50 FOR X = 1TO5
60 B$ = MID$(A$,X,1)
70 PLOT16,11 + X,B$
80 NEXT X
90 END
```

La orden HIRES le permite al usuario dar entrada a la modalidad en alta resolución del Oric. En la modalidad HIRES la pantalla tiene su punto de origen en la esquina superior izquierda.

En el BASIC del Oric existen varias órdenes que atañen a los gráficos: CURSETx,y,k posiciona el cursor sobre el punto dado por las coordenadas (x,y). El tercer número, "k", permite emplear con CURSET distintas funciones.

Valor de k	Función
0	traza el pixel en el color del fondo
1	traza el pixel en el color del primer plano
2	invierte los colores
3	no hace nada

CURMOVx,y,k es similar a CURSET, con la excepción de que el movimiento del cursor está en función de su posición anterior. DRAWx,y,k dibuja una línea recta desde la posición de partida del cursor hasta un punto "x" unidades a través e "y" unidades hacia arriba. CIRCLEr,k es una orden que traza en la pantalla un círculo de radio "r". PATTERN es una orden interesante y poco usual: trocea los círculos y las líneas dibujados en una serie de puntos o rayas. El patrón exacto se define mediante el número "n", cuya escala va de 0 a 255. El Oric toma este número y se vale del patrón de bits de su equivalente binario para producir un patrón repetitivo de puntos, rayas o espacios. He aquí dos ejemplos que ilustran su utilización:

Valor de n	Equivalente binario	Patrón producido
170	10101010	- - - - - - - -
15	00001111	— —

Por último, está la orden FILLa,b,n. Cada fila en los espacios de caracteres de la pantalla del Oric posee un número relacionado con ella que alude a los colores del fondo y del primer plano, al carácter presente y si el carácter está intermitente o no. Se dice que este número es el *atributo* de esa fila. FILLa,b,n rellena las celdas del carácter "b" por "a" filas con los atributos representados por el número "n".

```
10 REM CONO
20 HIRES
30 CURSET120,0,3
40 PAPER3:INK4
50 FOR R = 1TO65
60 PATTERN 200-R
70 CURMOV0,2,3
80 CIRCLE R,1
90 NEXT R
100 END
```

PATTERN de un cono
Este programa demuestra alguna de las capacidades del Oric-1 en alta resolución. Se dibuja una forma cónica empleando un conjunto de círculos de radio creciente. Observe, asimismo, la utilización de la orden PATTERN para fragmentar las circunferencias a medida que se van dibujando

Agdpyp w bcqagdpyp

La criptografía fue una de las primeras aplicaciones de los ordenadores. En la actualidad, elaborar y descifrar un código sencillo está al alcance de un programador de BASIC

Toda nuestra comunicación con los demás está codificada. Tanto el habla como el lenguaje escrito son inteligibles sólo si la persona que recibe el mensaje conoce el código del comunicante. Lo mismo sucede con nuestras conversaciones con los ordenadores. La mayoría de los ordenadores personales se comunican por medio de una versión del BASIC para ser accesibles a la mayoría de la gente, pero nosotros sabemos que la propia máquina no emplea este lenguaje para realizar sus funciones: ella debe primero interpretar las sentencias en BASIC en una forma puramente numérica que después utiliza para establecer las secuencias de conmutación definidas en el programa y, de este modo, producir los resultados deseados. Los códigos de este tipo (lenguajes humanos y lenguajes de programación) son de fácil acceso en nuestra vida cotidiana. Con un poco de esfuerzo y voluntad, cualquiera puede aprender francés, alemán, BASIC o FORTRAN.

Compresión de datos

Los usuarios de ordenadores que necesitan almacenar grandes cantidades de archivos de texto están a la búsqueda constante de formas de comprimir los datos en aquellos archivos. Una forma de conseguirlo es la distintivación. De modo muy similar a la serie de microordenadores ZX de Sinclair que producen una palabra completa reservada en BASIC al digitar una sola tecla, un distintivo se puede sustituir por una palabra o una frase.
Además, las técnicas de codificación también se utilizan para comprimir aún más los datos. Se considera que Compact, una utilidad Unix, puede comprimir archivos de textos en un 38 %, y Clip, que funciona bajo CP/M, consigue regularmente resultados aún mejores. Compactor, que se ejecuta en el Commodore 64, realiza la misma función para programas en BASIC mediante la eliminación de sentencias REM, espacios innecesarios, etc.

Pero existe otro tipo de codificación (o *criptografía*, para usar la palabra adecuada) que tiene por objetivo exactamente lo contrario de la comunicación: su finalidad consiste en evitar que lo comprendan todos, a excepción de un reducido grupo al cual está destinado el mensaje. Hasta la segunda mitad del siglo XX, la transmisión de información en una forma ininteligible para el público general era privativa de los gobiernos y de algún que otro asunto industrial importante. Pero más recientemente, la criptografía se ha convertido en algo cotidiano.

Las claves y los códigos oscilan desde los muy simples (la suma o la resta de un valor determinado a cada byte, o la sustitución según algún formato de un carácter por otro cada vez que éste aparece) hasta las claves en extremo complejas por las que se encaminan los más recientes avances de la teoría de los números. Estas claves no contienen ningún elemento de repetición y, por consiguiente, no son descifrables con los tradicionales métodos de decodificación por análisis de frecuencia.

Quizás la más sencilla de todas las técnicas significativas de criptografía sea la *clave del César* (que probablemente se utilizó por primera vez en la época del Imperio romano). Para descifrar la clave del César sólo se necesita el mensaje y un conocimiento de la clave, de modo que no hay que consultar voluminosos libros de códigos ni documento alguno, ni se necesitan unas máquinas especiales. He aquí un breve mensaje escrito en clave del César:

AYJYZNXNQ W BPYENLCQ

Podemos aventurar algunas suposiciones acerca de estas palabras crípticas, a tenor de la forma en que se separan los grupos cifrados (aunque, por supuesto, ¡esto se podría haber hecho para crear mayor confusión!). Lo más obvio que destaca a primera vista es que el mensaje consta de tres palabras: la primera posee 9 letras, la segunda posee 1 y la última, 8. También es claro que la primera y la tercera palabras terminan con la misma letra. Aquí la última letra final común (Q) es, asimismo, una de las tres letras del mensaje que se repiten con mayor frecuencia (las otras dos son la Y y la N). Para el criptoanalista esta observación tiene un valor considerable (al menos, cuando sabe en qué idioma está trabajando). En castellano, las letras que se presentan con mayor frecuencia son la A y la E entre las vocales, y la S entre las consonantes; esta última, dado que con ella se forma el plural, suele hallarse al final de las palabras.

Con una muestra tan reducida como la que tenemos aquí (un total de sólo 18 letras, cifra que todo estadístico consideraría muy insuficiente para basar en ella cualquier análisis), es probable que nuestros resultados sean falibles. Pero, aun así, vamos a probar con la sustitución de frecuencia y veremos si los resultados obtenidos tienen algún significado. En primer lugar vamos a reemplazar la Q por la S, por ser aquélla la última letra de las dos palabras más largas.

AYJYZNXNs W BPYENLCs

El mensaje aún no tiene significado, pero existen otras pistas. ¿Qué hay de la relación entre la letra original y la letra por la cual la hemos sustituido? En el alfabeto, la Q está dos lugares antes que la S. ¿Qué sucedería si sometiéramos el resto del mensaje a la misma transformación? Dos lugares después de la Y (la otra de nuestras letras que aparece con mayor frecuencia) está la A (si consideramos el alfabeto como una cadena ininterrumpida), así que vamos a intentar agregar esta información:

AaJaZNXNs W BPaENLCs

En la primera palabra tenemos ahora dos vocales intercaladas, que en castellano es una construcción válida. Además, la letra final es una S, hecho que en esta lengua es de común ocurrencia, de modo que tal vez estemos en el camino correcto. Vamos a

someter el resto del mensaje a la misma transformación. Dos lugares tras de la A se encuentra la C; dos lugares después de la J está la N; aplicando la misma correspondencia, la Z se convierte en B, la N en O y la X en Z... De esta manera llegamos a la solución: *Calabozos y dragones (Dungeons and dragons)*, un atractivo juego de aventuras.

La clave del César es, pues, un código de sustitución que se basa en "deslizar" el alfabeto hacia atrás o hacia adelante según un número que es secreto y que nos dará el nuevo valor de cada carácter. Sin utilizar las frecuencias, el mensaje puede simplemente construirse según una serie clave de

La clave del César

Este programa (escrito en BASIC Commodore) codificará textos en clave del César utilizando una serie de claves múltiples de cinco elementos. El mensaje aparece en texto llano mientras se le va dando entrada, y cuando se pulsa RETURN se imprime la versión cifrada. Al mensaje se le debe dar entrada sin espacios ni signos de puntuación

```
10 INPUT "DE ENTRADA A UNA CLAVE DE CINCO
   CIFRAS";K$
20 INPUT "DE ENTRADA AL MENSAJE";M$
30 FOR I = 1 TO LEN(M$)
40 LET J = I — INT (I/5)*5 + 1
50 REM***ROTA A TRAVES DE LA CLAVE
60 LET M = ASC(MID$(M$,1,1)) — VAL(MID$
   (K$,J,1))
70 IF M < 65 THEN LET M = M + 26
80 PRINT CHR$(M);
90 NEXT I
```

Para el Spectrum, se debe reemplazar la línea 60 por:

```
60 LET M = CODE(M$(I)) — VAL(K$(J))
```

transformaciones: 24225, por ejemplo. En este caso la primera letra se desplazaría dos lugares, la segunda cuatro, la tercera dos y así sucesivamente. Cuando se llega al último número de la serie clave la volvemos a iniciar. Con esta serie clave, el mensaje de muestra *Calabozos y dragones* quedaría:

AWJYWNVNQ T BÑYEKLAQ

En este caso, el análisis de frecuencia sería inútil, pues no hay uniformidad en la sustitución: una letra tendrá distintas sustitutas según su posición en el mensaje global. Otra clave sencilla autocontenida da el mismo mensaje en estos términos:

CBSDOAAOO RGNSLZYAE

Si observamos atentamente, podemos ver que esta serie de caracteres es en realidad un anagrama de *Calabozos y dragones*, completo y con los dos espacios entre las palabras. Aquí simplemente estamos tratando de determinar el algoritmo criptográfico, a partir de ejemplos tanto de texto llano como de texto cifrado, un procedimiento sorprendentemente común. Si la clave ha de ser comprensible al destinatario del mensaje, entonces la combinación de las letras debe ser predecible de alguna forma. Esta clave particular, conocida como *bar fence*, también requiere que el decodificador la conozca; en este caso es 3. Vamos a tomar los cinco primeros caracteres y escribirlos con tres espacios de por medio:

C***B***S * D***O

¿Reconoce algo? Entonces probemos con esto: escriba el mensaje de texto llano en tres líneas, yendo hacia arriba y hacia abajo entre ellas, de la siguiente manera:

```
C       B       S       D       O
  A   A   O   O   *   *   R   G   N   S
    L       Z       Y       A       E
```

Los asteriscos representan los espacios entre las palabras; el método criptográfico es llano.

Los ejemplos que hemos citado hasta el momento han sido todos cifras, definidas como un método de escritura secreta que utiliza la sustitución o transformación de letras de acuerdo a una clave. Los códigos son algo diferentes, en el sentido de que tienden a reemplazar bloques enteros por otros bloques, normalmente más pequeños (permitiendo, de este modo, al mismo tiempo, comprimir los datos). Su inconveniente es que exigen que ambas partes posean un libro de código para que se puedan comunicar mensajes. Un ejemplo de esta técnica emplea una novela, un periódico u otro texto que se pueda conseguir con facilidad, e indica las palabras del mensaje dando el número de secuencia en que se producen. Un texto como:

> "Maigret leyó con atención el aviso: 'Por favor, al objeto de favorecer el tráfico, permanezca al volante, no baje del coche'. Sonrió. Ahora comprendía por qué en su momento el cadáver de Smith no fue identificado."

podría ser la clave para el código 4,6,7,17,2. Quizá usted pueda descifrar el alarmante mensaje...

Un ordenador de cualquier clase puede ser de enorme valor al intentar cifrar o descifrar mensajes criptográficos. Un requisito primordial de la clave del César, por ejemplo, es la capacidad de desplazamiento a través de una serie alfanumérica, sumando o restando una variable al valor ASCII de cada carácter, que se puede entonces imprimir. Dicha constante ha de ser susceptible de modificarse cuando se ejecuta el programa, y debe hacer que el alfabeto se "enrolle" (es decir, una A, cuando la clave fuera uno, debería dar Z). Así que:

AOPK AO PKZK WIECK



Criptoanálisis
Una de las primeras utilizaciones del ordenador fue la de romper los códigos de sustitución de claves múltiples, muy complicados, que empleaban ambos bandos durante la segunda guerra mundial. Los alemanes habían preparado una máquina, denominada ENIGMA, que generaba sus propias claves. Los criptogramas resultantes, sumamente complicados, obligaban a los aliados a dedicar muchísimos esfuerzos a interpretarlos. El grupo del Colossus, que trabajaba en Bletchley Park y del que Alan Turing era miembro destacado, obtuvo finalmente el éxito

Cortesía Science Museum

Cuestión de estilo

Conocidas las reglas fundamentales del BASIC, analicemos importantes aspectos del estilo de programación y algunas órdenes nuevas para perfeccionar la técnica de programación

El programa que hemos venido construyendo en los últimos capítulos del curso para la agenda de direcciones computerizada recurre a muchas de las configuraciones más importantes del lenguaje BASIC, pero no a todas. En los próximos capítulos veremos lo que el BASIC le ofrecerá si es que desea convertirse en un programador de nivel avanzado.

Programas en lenguaje máquina

La mayoría de las versiones de BASIC permiten incluir como parte del programa rutinas escritas en lenguaje máquina. En líneas generales, existen dos maneras de hacerlo. La más sencilla consiste en utilizar PEEK y POKE. La sentencia PEEK se emplea para examinar direcciones de memoria específicas. Por ejemplo, LET X = PEEK(1000) obtendrá el valor almacenado en la posición de memoria 1000 y se lo asignará a la variable X. Al ejecutar PRINT X se imprimirá, entonces, el valor que estaba (y que sigue estando) en la posición 1000. He aquí un breve programa para "mirar" (PEEK) qué contienen 16 posiciones de memoria e imprimirlas en la pantalla:

```
10 INPUT "DEME DIRECCION INICIAL DE 'PEEK'";S
20 PRINT
30 FOR L = 1 TO 16
40 LET A = PEEK(S)
50 PRINT "LA POSICION ";S;" CONTIENE: ";A
60 LET S = S + 1
70 NEXT L
80 PRINT "PULSE BARRA ESPACIADORA PARA
   EXAMINAR LAS 16 POSICIONES SIGUIENTES"
90 PRINT "O RETURN PARA TERMINAR"
100 FOR I = 1 TO 1
110 LET C$ = INKEY$
120 IF C$ <> CHR$(13) AND C$<> " " THEN
    I = 0
130 NEXT I
140 IF C$ = CHR$(13) THEN GOTO 160
150 GOTO 30
160 END
```

El bucle desde la línea 100 a la 130 verifica la entrada desde el teclado y después va, o bien al final del programa, si la tecla pulsada fue un RETURN (13, en ASCII), o bien al principio, saltándose INPUT.

Si se desea, también es posible imprimir el carácter ASCII situado en una posición de memoria: basta usar PRINT CHR$(A). Pero cuidado con las sorpresas, porque los valores ASCII inferiores a 32 (que es el "espacio" en ASCII) no están definidos uniformemente. Todos los valores ASCII desde 0 a 31 representan caracteres no susceptibles de impresión o funciones especiales, como los movimientos del cursor. Los diversos fabricantes de ordenadores prácticamente sólo coinciden en que ASCII 13 es por lo general retorno del carro y que ASCII 7 hace sonar el altavoz interno o produce un pitido.

POKE es lo inverso de PEEK. Permite escribir cualquier valor entre 0 y 255 en cualquier posición de memoria RAM. Sin embargo, esta facilidad se debe emplear con extrema cautela, porque escribir en una parte de la memoria qué ya esté siendo utilizada por el programa puede producir resultados inesperados y hasta ruinosos. Las rutinas escritas en código de lenguaje máquina se pueden "colocar" (POKE) en las direcciones apropiadas y cuando se ejecuta el programa se las puede invocar mediante la sentencia CALL. Cómo escribir programas en código de lenguaje máquina es un tema que está fuera de los límites de este curso. Baste decir que el código de lenguaje máquina se ejecuta muchísimo más rápido que las mejores versiones de BASIC. En aquellas situaciones en las que la velocidad de ejecución es esencial, o en las que se requiere gran precisión, el código de lenguaje máquina es, con mucho, la mejor alternativa.

Mover el cursor

Ahora muchos ordenadores personales permiten dirigirse hacia un punto de la pantalla de inmediato, pero aun cuando su máquina no lo admita, se puede mover el cursor a izquierda, derecha, arriba y abajo de la pantalla con relativa facilidad. Primero necesita usted saber qué códigos ASCII se utilizan para representar las teclas de control del cursor. El breve programa que ofrecemos a continuación, tras pulsar una tecla le informará acerca del valor ASCII correspondiente a dicha tecla:

```
1 REM HALLA LOS CODIGOS ASCII PARA LAS
  TECLAS DEL CURSOR
10 PRINT "PULSE UNA TECLA";
20 FOR I = 1 TO 1
30 LET K$ = INKEY$
40 IF K$ = "" THEN I = 0
50 NEXT I
60 PRINT ASC(K$)
70 GOTO 10
80 END
```

Esta rutina le permitirá también hallar el código para la tecla RETURN (por lo general, 13), ESC (comúnmente, 27) y la tecla para espacio (por lo general, 32), además de los códigos para las teclas de control del cursor. El ordenador Sord M23, en el cual se probaron todos los programas para este curso de programación BASIC, utiliza los valores 8 para cursor a la izquierda, 28 para cursor a la derecha, 29 para cursor arriba y 30 para cursor abajo. Es probable que su ordenador exhiba valores diferentes. Sustituyendo los valores que ha hallado

para los códigos de control del cursor de su ordenador en el programa anterior, pruebe con éste:

```
10 PRINT CHR$(12): REM USAR CLS O CODIGO
   ADECUADO
20 FOR L = 1 TO 39
30 PRINT "*";
40 NEXT L
50 FOR L = 1 TO 22
60 PRINT CHR$(8);:REM USAR CODIGO 'CURSOR
   IZQUIERDA'
70 NEXT L
80 FOR L = 1 TO 4
90 PRINT "@";
100 NEXT L
110 END
```

Con esto debería imprimirse en la pantalla una línea parecida a ésta:

*****************@@@@******************

Las sentencias de la 20 a la 40 simplemente habrán impreso una línea de 39 estrellas, y las sentencias 50 a 70 "imprimieron" el "carácter" del cursor izquierda 22 veces, de modo que el cursor se desplazó 22 lugares hacia atrás a lo largo de la línea. De la 80 a la 100 imprimieron luego @ cuatro veces y entonces el programa terminó. Las técnicas de programación como ésta permiten que el programador desplace el cursor alrededor de la pantalla para imprimir nuevos caracteres en nuevas posiciones que podrían no conocerse hasta que se calcularan los valores en el programa. Esta técnica ofrece la ventaja de posibilitar la utilización de caracteres de pantalla normales para trazar gráficos sencillos, sin necesidad de recurrir a las configuraciones especiales para gráficos del ordenador (si posee alguna).

Para ver cómo emplear este tipo de control del cursor para producir gráficos como una salida de sus programas, pruebe con este programa:

```
10 PRINT "ESTE PROGRAMA IMPRIME UN GRAFICO
   DE BARRAS DE 3 VARIABLES"
20 INPUT "DE ENTRADA A LOS TRES
   VALORES";X,Y,Z
30 PRINT
40 FOR L = 1 TO 2
50 FOR A = 1 TO X
60 PRINT "*";
70 NEXT A
80 PRINT CHR$(13)
90 NEXT L
100 FOR L = 1 TO 2
110 FOR A = 1 TO Y
120 PRINT "+";
130 NEXT A
140 PRINT CHR$(13)
150 NEXT L
160 FOR L = 1 TO 2
170 FOR A = 1 TO Z
180 PRINT "#";
190 NEXT A
200 PRINT CHR$(13)
210 NEXT L
220 PRINT
230 END
```

El programa imprime un diagrama de barras horizontales de las tres variables. Las barras se imprimen de izquierda a derecha siguiendo el movimiento "natural" del cursor. Observe que en las sentencias 80, 140 y 200 se necesita una PRINT CHR$(13). Ello se debe a que los puntos y coma al final de las sentencias PRINT suprimen los retornos del carro (13 es el código ASCII para <CR>).

En torno a las variables

Hasta ahora hemos tratado las variables como si sólo fueran de dos tipos (numéricas y alfanuméricas). En realidad, el BASIC conoce diversos tipos de variables numéricas, y un buen programador siempre especificará el tipo correcto para economizar memoria y asegurar la validez.

Cuando se enuncia una variable en un lenguaje de programación, automáticamente se reserva cierta cantidad de memoria para almacenar dicha variable. Si el programa sabe que la variable será siempre un entero (p. ej., LET NO = VOTOS — (SI + NULOS) se necesita reservar menos memoria para la variable. Si tenemos una variable que puede asumir un número indefinido de valores distintos (p. ej., LET AREA = PI * RADIO * RADIO), la cantidad de espacio de memoria a destinar será mayor.

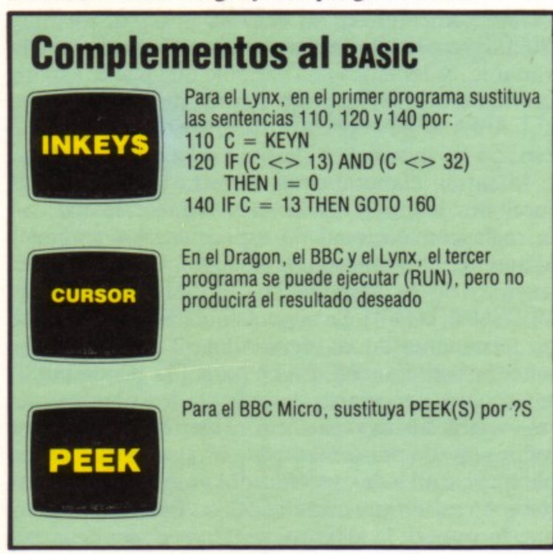
Mientras elaborábamos nuestra agenda de direcciones computerizada, nos familiarizamos con la regla convencional de especificar las variables alfanuméricas utilizando el signo $ después del nombre de la variable (p. ej., LET BUSCLV$ = MODCAM$(TAMA)). Se suponía que las variables que no terminaran en un signo "dólar" eran variables numéricas ordinarias. Sin embargo, se pueden utilizar acuerdos similares después de los nombres de las variables para especificar el tipo de variable numérica. Se da por sentado que un nombre de variable que carezca de especificador es una variable numérica real de precisión sencilla. Entre los signos que reconocen la mayoría de las versiones de BASIC se incluyen: % para especificar una variable de números enteros, ! para especificar una variable de precisión sencilla, y # para especificar una variable de precisión doble (es decir, la variable puede almacenar el doble de dígitos significativos). A continuación ofrecemos el fragmento de un programa hipotético que recurre a estos signos:

```
70 LET JUGADOR $ = "LUIS": REM VARIABLE
   ALFANUMERICA
80 LET PUNTUACION % = 0: REM VARIABLE DE
   ENTEROS
90 LET PI! = 3,1416:REM VARIABLE DE
   PRECISION SENCILLA
100 LET AREA # = PI*R*R: REM VARIABLE DE
    PRECISION DOBLE
110 LET INTENTO = 6: REM SE SUPONE QUE ES
    REAL DE PRECISION SENCILLA
```

Es necesario señalar que no todas las versiones de BASIC admiten todos estos tipos de variables. El Spectrum, por ejemplo, no posee variables de enteros. Los enteros se almacenan simplemente como números reales de precisión sencilla. Tampoco admite números de precisión doble. No obstante, en el BASIC Spectrum los números de precisión sencilla se calculan con nueve guarismos significativos, contra los sólo siete guarismos significativos del BASIC Microsoft. El BBC Micro admite variables del tipo entero y los reales de precisión sencilla se calculan en nueve guarismos significativos. El BASIC Microsoft admite variables de precisión doble con 16 lugares significativos.

Los ordenadores que aceptan variables de enteros normalmente destinan dos bytes para almacenar un número, que puede estar dentro del intervalo [−32 768 , 32 767]. Por lo general este intervalo es perfectamente adecuado para variables tales como puntuaciones, número de empleados, contadores de bucles FOR...NEXT y otras cantidades que sólo admiten enteros. Dado que para almacenar el número sólo se emplean dos bytes, la utilización de variables de enteros, si se dispone de ellas, supondrá un ahorro de memoria, si bien en muchas versiones de BASIC ello sólo es válido para las matrices de enteros y no para las variables individuales.

En el próximo capítulo de este curso de programación sopesaremos las ventajas y las desventajas del BASIC como lenguaje de programación.



La agenda de direcciones para el Spectrum
Ésta es la versión completa para el Spectrum del programa para la agenda de direcciones. En el próximo capítulo incluiremos los "Complementos al BASIC" correspondientes al Lynx, Dragon 32, BBC Micro, Commodore 64 y Vic-20, que harán referencia a este listado

```
1 REM *CREAR ARCHIVO DE DATOS*
2 DIM N$(50,30)
3 LET N$(1)="@VACIO"
4 INPUT"INSERTE CINTA DE DATOS, PULSE PLAY Y PULSE
  'ENTER'";A$
5 SAVE "NCAM" DATA N$()
6 INPUT"REBOBINE CINTA, PULSE PLAY, PULSE 'ENTER'";A$
7 VERIFY "NCAM" DATA N$()
8 STOP
```

Éste es el programa de inicialización que crea la matriz en cinta por primera vez. Después de ejecutar este programa, rebobine la cinta de datos, cargue (LOAD) el programa principal (listado inferior) y ejecútelo (RUN). Usted no volverá a necesitar el programa de inicialización otra vez, a menos que desee crear un nuevo archivo para la agenda de direcciones.

```
10 REM 'PROPRI'
20 REM *INICIL*
30 GOSUB 1000
40 REM *PRESEN*
50 GOSUB 3000
60 FOR M=1 TO 1
70 LET M=0
80 REM *ELECCN*
90 GOSUB 3500
100 REM *EJECUT*
110 GOSUB 4000
120 IF OPCN=9 THEN LET M=1
130 NEXT M
140 STOP

1000 REM S/R *INICIL*
1010 GOSUB 1100
1020 GOSUB 1400
1030 GOSUB 1600
1090 RETURN

1100 REM S/R *CREMAT*
1110 DIM N$(50,30)
1120 DIM M$(50,30)
1130 DIM C$(50,30)
1140 DIM D$(50,15)
1150 DIM P$(50,15)
1160 DIM T$(50,15)
1170 DIM X$(50,30)
1180 DIM B$(30) :DIM Z$(30)
1190 DIM U$(30):DIM W$(15)
1210 LET TAMA=0
1220 LET RMOD=0
1230 LET CLAR=1
1240 LET CURS=0
1250 LET Z$="@VACIO"
1260 LET Q$=B$
1300 RETURN

1400 REM S/R *LEARCH*
1405 INPUT "INSERTE CINTA DE DATOS, PULSE PLAY Y PULSE
     'ENTER'";A$
1410 LOAD "NCAM" DATA N$()
1420 IF N$(1)=Z$ THEN LET Q$=Z$:RETURN
1430 LOAD "MCAM" DATA M$()
1440 LOAD "CCAM" DATA C$()
1450 LOAD "DCAM" DATA D$()
1460 LOAD "PCAM" DATA P$()
1470 LOAD "TELCAM" DATA T$()
1480 LOAD "INDCAM" DATA X$()
1485 INPUT"PARE LA CINTA Y PULSE 'ENTER'";A$
1490 REM 'TAMANO'
1500 LET TAMA=51
1510 FOR L=1 TO 50
1520 IF N$(L)=B$ THEN LET TAMA=L:LET L=50
1530 NEXT L
1540 RETURN

1600 REM S/R *ESBAND*
1640 IF Q$=Z$ THEN LET TAMA=1
1690 RETURN

3000 REM *PRESEN*
3010 CLS
3020 PRINT:PRINT:PRINT:PRINT
3060 PRINT TAB(7);"*BIEN VENIDO A LA*"
3070 PRINT TAB(5);"*AGENDA COMPUTERIZADA*"
3080 PRINT TAB(8);"*DE MI COMPUTER*"
3090 PRINT
3100 PRINT "(PULSE BARRA ESPAC. PARA SEGUIR)"
3110 FOR L=1 TO 1
3120 IF INKEY$<>" " THEN LET L=0
3130 NEXT L
3140 CLS
3150 RETURN

3500 REM S/R *ELECCN*
3520 IF Q$=Z$ THEN GOSUB 3860:RETURN
3540 REM 'IMMENU'
3550 CLS
3560 PRINT "SELECCIONE UNO DE LOS SIGUIENTES"
3570 PRINT:PRINT:PRINT
3600 PRINT"1. HALLAR REGISTRO (DE NOMBRE)"
3610 PRINT"2. HALLAR NOMBRES (DE NOMBRE INCOMPLETO)"
3620 PRINT"3. HALLAR REGISTRO (DE CIUDAD)"
3630 PRINT"4. HALLAR REGISTRO (DE INICIAL)"
3640 PRINT"5. LISTAR TODOS LOS REGISTROS"
3650 PRINT"6. AGREGAR REGISTRO NUEVO"
3660 PRINT"7. MODIFICAR REGISTRO"
3670 PRINT"8. BORRAR REGISTRO"
3680 PRINT"9. SALIR Y GUARDAR"
3690 PRINT:PRINT
3710 REM 'ASOPCN'
3750 PRINT"DE ENTRADA A OPCION (1-9)"
3760 FOR L=1 TO 1
3770 FOR I=1 TO 1
3780 LET A$=INKEY$
3790 IF A$="" THEN LET I=0
3800 NEXT I
3810 LET OPCN = CODE A$-48
3820 IF (OPCN<1) OR(OPCN>9) THEN LET L=0
3840 NEXT L
3850 RETURN

3860 REM S/R *PRIMERA*
3870 LET OPCN=6
3880 CLS
3890 PRINT
3900 PRINT TAB(6);"NO HAY REGISTROS EN"
3910 PRINT TAB(3);"EL ARCHIVO. DEBERA EMPEZAR"
3920 PRINT TAB(4);"POR AGREGAR UN REGISTRO"
3930 PRINT
3940 REM *CONTINUAR*
3950 GOSUB 3100
3990 RETURN

4000 REM S/R *EJECUT*
4040 IF OPCN=1 THEN GOSUB 5700
4050 REM 2 ES *ENCNOM*
4060 REM 3 ES *ENCCIU*
4070 REM 4 ES *ENCINI*
4080 REM 5 ES *LISREG*
4090 IF OPCN=6 THEN GOSUB 4200
4100 IF OPCN=7 THEN GOSUB 6600
4110 IF OPCN=8 THEN GOSUB 7500
4120 IF OPCN=9 THEN GOSUB 5000
4140 RETURN

4200 REM S/R *INCREG*
4210 CLS
4220 INPUT "DE ENTRADA AL NOMBRE";N$(TAMA)
4230 INPUT "DE ENTRADA A LA CALLE";C$(TAMA)
4240 INPUT "DE ENTRADA A LA CIUDAD";D$(TAMA)
4250 INPUT "DE ENTRADA A LA PROVINCIA";P$(TAMA)
4260 INPUT "DE ENTRADA AL NUMERO DE TELEFONO";T$(TAMA)
4270 LET RMOD=1:LET CLAR=0
4280 LET X$(TAMA)=STR$(TAMA)
4290 LET Q$=""
4300 GOSUB 4500
4310 LET OPCN=0
4320 LET TAMA=TAMA+1
4350 RETURN

4500 REM S/R *MODNOM*
4510 REM CONVERTIR A MAYUSCULAS
4520 LET R$=N$(TAMA):LET S$=""
4530 FOR L=1 TO LEN(R$)
```

```
4540 LET A$=R$(L)
4550 LET T=CODE A$
4560 IF T>=97 THEN LET T=T-32
4570 LET A$=CHR$ T
4580 LET S$=S$+A$
4590 NEXT L
4600 LET R$=S$:LET S$=" ":LET A$=" ":LET T=LEN(R$):LET S=0
4610 REM LOCALIZAR PRIMER ESPACIO
4620 FOR L=1 TO T
4630 IF R$(L)=" " THEN LET S=L:LET L=T
4640 NEXT L
4650 REM DEPURAR Y COLOCAR NOMBRE DE PILA EN S$
4660 FOR L=1 TO S-1
4670 IF CODE(R$(L))>64 THEN LET S$=S$+R$(L)
4680 NEXT L
4690 REM DEPURAR Y COLOCAR APELLIDO EN A$
4700 FOR L=S+1 TO LEN(R$)
4710 IF CODE(R$(L))>64 THEN LET A$=A$+R$(L)
4720 NEXT L
4730 LET M$(TAMA)=A$+" "+S$
4740 LET S$="":LET A$="":LET S=0
4750 RETURN

5000 REM S/R *SAPROG*
5010 IF (RMOD=0) AND (CLAR=1) THEN RETURN
5020 IF (RMOD=1) AND (CLAR=0) THEN GOSUB 5200
5030 GOSUB 5600
5040 RETURN

5200 REM S/R *CLSREG*
5210 FOR K=1 TO 1
5220 LET S=0
5230 FOR L=1 TO TAMA-2
5240 LET T=L+1
5250 IF M$(L)>M$(T) THEN GOSUB 5400
5260 NEXT L
5270 IF S=1 THEN LET K=0
5280 NEXT K
5290 LET CLAR=1
5300 RETURN

5400 REM *INTERCALAR*
5410 LET U$=N$(L):LET N$(L)=N$(T):LET N$(T)=U$
5420 LET U$=M$(L):LET M$(L)=M$(T):LET M$(T)=U$
5430 LET U$=C$(L):LET C$(L)=C$(T):LET C$(T)=U$
5440 LET W$=D$(L):LET D$(L)=D$(T):LET D$(T)=W$
5450 LET W$=P$(L):LET P$(L)=P$(T):LET P$(T)=W$
5460 LET W$=T$(L):LET T$(L)=T$(T):LET T$(T)=W$
5470 LET X$(L)=STR$(L)
5480 LET X$(T)=STR$(T)
5490 LET S=1
5500 RETURN

5600 REM S/R *GRDREG*
5605 INPUT"INSERTE CINTA DE GRABACION Y PULSE
     'ENTER'":A$
5610 SAVE "NCAM" DATA N$()
5620 SAVE "MCAM" DATA M$()
5630 SAVE "CCAM" DATA C$()
5640 SAVE "DCAM" DATA D$()
5650 SAVE "PCAM" DATA P$()
5660 SAVE "TELCAM" DATA T$()
5670 SAVE "INDCAM" DATA X$()
5680 INPUT"DETENGA LA CINTA Y PULSE 'ENTER'";A$
5690 RETURN

5700 REM S/R *ENCREG*
5710 CLS
5720 IF CLAR=0 THEN GOSUB 5200
5730 PRINT:PRINT
5740 PRINT TAB(6);"BUSCANDO UN REGISTRO"
5750 PRINT TAB(9);"POR EL NOMBRE"
5760 PRINT
5770 PRINT TAB(3);"DIGITE EL NOMBRE COMPLETO"
5780 PRINT TAB(1);"POR ORDEN NOMBRE APELLIDO"
5790 PRINT:PRINT
5800 INPUT"EL NOMBRE ES";N$(TAMA)
5810 GOSUB 4500
5820 LET U$=M$(TAMA)
5830 LET INF=1
5840 LET SUP=TAMA-1
5850 FOR X=1 TO 1
5860 LET MED=INT((INF+SUP)/2)
5870 IF M$(MED)<>U$ THEN LET X=0
5880 IF M$(MED)< U$ THEN LET INF=MED+1
5890 IF M$(MED)> U$ THEN LET SUP=MED-1
5900 IF INF>SUP THEN LET X=1
5910 NEXT X
5920 IF INF>SUP THEN LET CURS=0
5930 IF INF<=SUP THEN LET CURS=MED
5940 IF CURS=0 THEN GOSUB 6400:RETURN
5950 CLS
5960 PRINT
5970 PRINT TAB(7);"*HALLADO REGISTRO*"
5980 PRINT
5990 PRINT"NOMBRE:",N$(CURS)
6000 PRINT"CALLE:",C$(CURS)
6010 PRINT"CIUDAD:",D$(CURS)
6020 PRINT"PROVINCIA:",P$(CURS)
6030 PRINT"TELEFONO:",T$(CURS)
6040 PRINT
6050 PRINT"PULSE UNA LETRA PARA IMPRIMIR"
6060 PRINT"O BARRA ESPAC. PARA CONTINUAR"
6070 FOR I=1 TO 1
6080 LET A$=INKEY$
6090 IF A$="" THEN LET I=0
6100 NEXT I
6110 IF A$<>" " THEN GOSUB 6200
6120 RETURN

6200 REM S/R *LSTCUR*
6210 LPRINT
6220 LPRINT"NOMBRE",N$(CURS)
6230 LPRINT"CALLE",C$(CURS)
6240 LPRINT"CIUDAD",D$(CURS)
6250 LPRINT"PROVINCIA",P$(CURS)
6260 LPRINT"TELEFONO",T$(CURS)
6270 LPRINT:LPRINT
6280 RETURN

6400 REM S/R *NINREG*
6410 CLS
6420 PRINT TAB(5);"*NO HALLADO REGISTRO*"
6430 PRINT TAB(4);"*EN FORMA ";N$(TAMA);" *"
6440 PRINT
6450 REM 'CONTINUAR'
6460 GOSUB 3100
6470 RETURN

6600 REM S/R *MODREG*
6610 CLS
6620 PRINT:PRINT:PRINT
6630 LET E$=CHR$ 13
6640 PRINT "*PARA MODIFICAR UN REGISTRO*"
6650 PRINT "*PRIMERO LOCALICE ESE REGISTRO*"
6660 GOSUB 5720
6670 IF CURS=0 THEN RETURN
6680 PRINT
6690 PRINT TAB(7);"MODIFICAR NOMBRE ?"
6700 PRINT
6710 PRINT "PULSE 'ENTER' PARA NUEVO NOMBRE"
6720 PRINT "O BARRA ESPAC. PARA SIGU. CAMPO"
6730 FOR I=1 TO 1
6740 LET A$=INKEY$
6750 IF (A$<>E$) AND (A$<>" ") THEN LET I=0
6760 NEXT I
6770 IF A$=E$ THEN INPUT"NUEVO NOMBRE";N$(CURS)
6780 IF A$=E$ THEN LET RMOD=1
6790 IF A$=E$ THEN LET CLAR=0
6800 IF A$=E$ THEN LET N$(TAMA)=N$(CURS)
6810 IF A$=E$ THEN GOSUB 4500
6820 IF A$=E$ THEN LET M$(CURS)=M$(TAMA)
6830 PRINT
6840 PRINT TAB(7);"MODIFICAR CALLE ?"
6850 PRINT
6860 PRINT "PULSE 'ENTER' PARA NUEVA CALLE"
6870 PRINT "O BARRA ESPAC. PARA SIGU. CAMPO"
6880 FOR I=1 TO 1
6890 LET A$=INKEY$
6900 IF (A$<>E$) AND (A$<>" ") THEN LET I=0
6910 NEXT I
6920 IF A$=E$ THEN LET RMOD=1
6930 IF A$=E$ THEN INPUT"NUEVA CALLE";C$(CURS)
6940 PRINT
6950 PRINT TAB(7);"MODIFICAR CIUDAD ?"
6960 PRINT
6970 PRINT "PULSE 'ENTER' PARA NUEVA CIUDAD"
6980 PRINT "O BARRA ESPAC. PARA SIGU. CAMPO"
6990 FOR I=1 TO 1
7010 LET A$=INKEY$
7020 IF (A$<>E$) AND (A$<>" ") THEN LET I=0
7030 NEXT I
7040 IF A$=E$ THEN LET RMOD=1
7050 IF A$=E$ THEN INPUT"NUEVA CIUDAD";D$(CURS)
7060 PRINT
7070 PRINT TAB(5);"MODIFICAR PROVINCIA ?"
7080 PRINT
7090 PRINT "PULSE 'ENTER' PARA NUEVA PROV."
7100 PRINT "O BARRA ESPAC. PARA SIGU. CAMPO"
7110 FOR I=1 TO 1
7120 LET A$=INKEY$
7130 IF (A$<>E$) AND (A$<>" ") THEN LET I=0
7140 NEXT I
7150 IF A$=E$ THEN LET RMOD=1
7160 IF A$=E$ THEN INPUT"NUEVA PROVINCIA";P$(CURS)
7170 PRINT
7180 PRINT "MODIFICAR NUMERO DE TELEFONO ?"
7190 PRINT
7200 PRINT "PULSE 'ENTER' PARA NUEVO TELEF."
7210 PRINT "O BARRA ESPACIADORA PARA SIGU. CAMPO"
7220 FOR I=1 TO 1
7230 LET A$=INKEY$
7240 IF (A$<>E$) AND (A$<>" ") THEN LET I=0
7250 NEXT I
7260 IF A$=E$ THEN LET RMOD=1
7270 IF A$=E$ THEN INPUT"NUEVO NUMERO";T$(CURS)
7280 RETURN

7500 REM S/R *BORREG*
7510 CLS
7520 PRINT:PRINT:PRINT:PRINT
7530 LET E$=CHR$ 13
7540 PRINT TAB(3);"*PARA BORRAR UN REGISTRO*"
7550 PRINT TAB(1);"*LOCALICE PRIMERO REG. DESEADO*"
7560 GOSUB 5720
7570 IF CURS=0 THEN RETURN
7580 PRINT
7590 PRINT "DESEA BORRAR ESTE REGISTRO ?"
7600 PRINT "*AVISO* -  NO HAY SEGUNDA OPORTUNIDAD !"
7610 PRINT
7620 PRINT TAB(3);"PULSE 'ENTER' PARA BORRAR"
7630 PRINT TAB(1);"O BARRA ESPAC. PARA CONTINUAR"
7640 FOR I=1 TO 1
7650 LET A$=INKEY$
7660 IF (A$<>E$) AND (A$<>" ") THEN LET I=0
7670 NEXT I
7680 IF A$=" " THEN RETURN
7690 FOR L=CURS TO TAMA-2
7700 LET T=L+1
7710 LET N$(L)=N$(T)
7720 LET M$(L)=M$(T)
7730 LET C$(L)=C$(T)
7740 LET D$(L)=D$(T)
7750 LET P$(L)=P$(T)
7760 LET T$(L)=T$(T)
7770 LET X$(L)=X$(T)
7780 NEXT L
7790 LET RMOD=1
7800 LET TAMA=TAMA-1
7810 RETURN
```

Un reto universitario

El primer ordenador programable del mundo se ideó en la Universidad de Manchester

Acabada la segunda guerra mundial, la Universidad de Manchester nombró a dos nuevos profesores. Max Newman fue nombrado profesor de matemáticas después del trabajo de descifrar códigos que realizó en Bletchley Park con el Colossus, el primer ordenador electromecánico del mundo, y el ingeniero de radares, F. C. Williams, obtuvo el puesto de ingeniería eléctrica. Williams se llevó consigo a un joven ayudante, Tom Kilburn, que se había familiarizado con los problemas de dispositivos de impulsos con memoria electrónica mientras trabajaba con los radares durante la guerra. Posteriormente, Kilburn se convertiría en el primer profesor de la recién creada cátedra de estudios informáticos en esa Universidad.

Durante un viaje realizado en 1946 a Estados Unidos de visita a las instalaciones de radares, Williams había visto el prototipo del ordenador de válvulas, ENIAC (véase p. 46) y, de regreso a Gran Bretaña, consiguió que la Royal Society invirtiera 35 000 libras esterlinas en un "laboratorio de máquinas de calcular" en la Universidad de Manchester. Éste no era el único centro docente empeñado en construir un ordenador de programas almacenados. La Universidad de Pennsylvania estaba construyendo el EDVAC, en la Universidad de Cambridge se estaba trabajando en el EDSAC y en el National Physical Laboratory se seguía trabajando en la elaboración del ACE (véase p. 88). No obstante, en todos estos proyectos se estaba utilizando un almacenamiento de memoria construido a partir de tubos en línea de demora de mercurio. El equipo de Manchester iba a hacer su máquina con un dispositivo de memoria, invención del propio Williams, que utilizaba un tubo de rayos catódicos. Para el otoño de 1947, Williams había conseguido retener 2 048 bits durante varias horas.

Gracias al "tubo Williams", en junio de 1948 el ordenador Manchester Mark I ejecutó con éxito un programa, convirtiéndose en el primer ordenador del mundo con programa almacenado. El Mark I podía ejecutar una instrucción en 1,2 milésimas de segundo. Un tubo de rayos catódicos permitía almacenar la información, la memoria tenía la ventaja de ser de acceso directo y se podía representar visualmente el contenido del almacenamiento principal o del registro de control.

Una vez demostrada la viabilidad del uso de un "tubo Williams" para el almacenamiento de memoria, se construyó una versión mejorada del Mark I, que se aplicó en problemas del diseño de óptica y en la generación de números primos. El consejero científico del Estado, sir Ben Lockspeiser, quedó tan impresionado con el rendimiento del ordenador, que encargó la construcción de una versión comercial del Mark I a una empresa local de Manchester. El Ferranti Mark I salió al mercado en febrero de 1951, anticipándose en cinco meses al UNIVAC; fue, pues, el primer ordenador disponible comercialmente.

Una importante innovación del Ferranti Mark I era su capacidad para modificar las instrucciones durante el proceso en virtud de otro almacenamiento denominado tubo "B". En el momento requerido, éste podía agregar su contenido al registro de control y, por consiguiente, modificar el código de la instrucción original. Este principio aceleraba la ejecución de los programas. En sus primeros ordenadores la IBM utilizó algunas de las patentes del Manchester, y en una visita a la oficina central de la sociedad, en Nueva York, blasonada por doquier con el lema THINK (PIENSA) de la empresa, se le preguntó a Williams cómo fue que el equipo de Manchester había logrado construir un ordenador cuando todos los esfuerzos de la IBM habían fracasado. Williams respondió sin titubear: "¡Es que nosotros no nos paramos a pensarlo tanto!"

La llegada de Alan Turing (véase p. 200) a Manchester, en 1948, fue un gran estímulo para las actividades de programación. En 1950, Turing sacó a la luz el primer manual de programación de Manchester. Dos años después, al equipo de Manchester ya le rondaba la idea de construir un ordenador más compacto y económico. Sus planes se aceleraron con la invención del transistor, y en noviembre de 1953 entraba en funcionamiento en Manchester el primer ordenador de transistores del mundo.

A fines de la década de los cincuenta, Estados Unidos se estaba adelantando en cuanto a la tecnología de ordenadores, a resultas de lo cual el gobierno británico decidió invertir en un proyecto que ayudaría a que Gran Bretaña recuperara su liderazgo. En diciembre de 1962 se encargó el ordenador Atlas, que se construyó bajo la dirección de Tom Kilburn. Empleaba una palabra de 48 bits con formato de dirección sencillo, un almacenamiento principal de 16 Kbytes y una memoria en tambor de lectura solamente de ocho Kbytes. Se vendieron unidades al Atomic Energy Research Establishment, en Harwell, y a la British Petroleum.

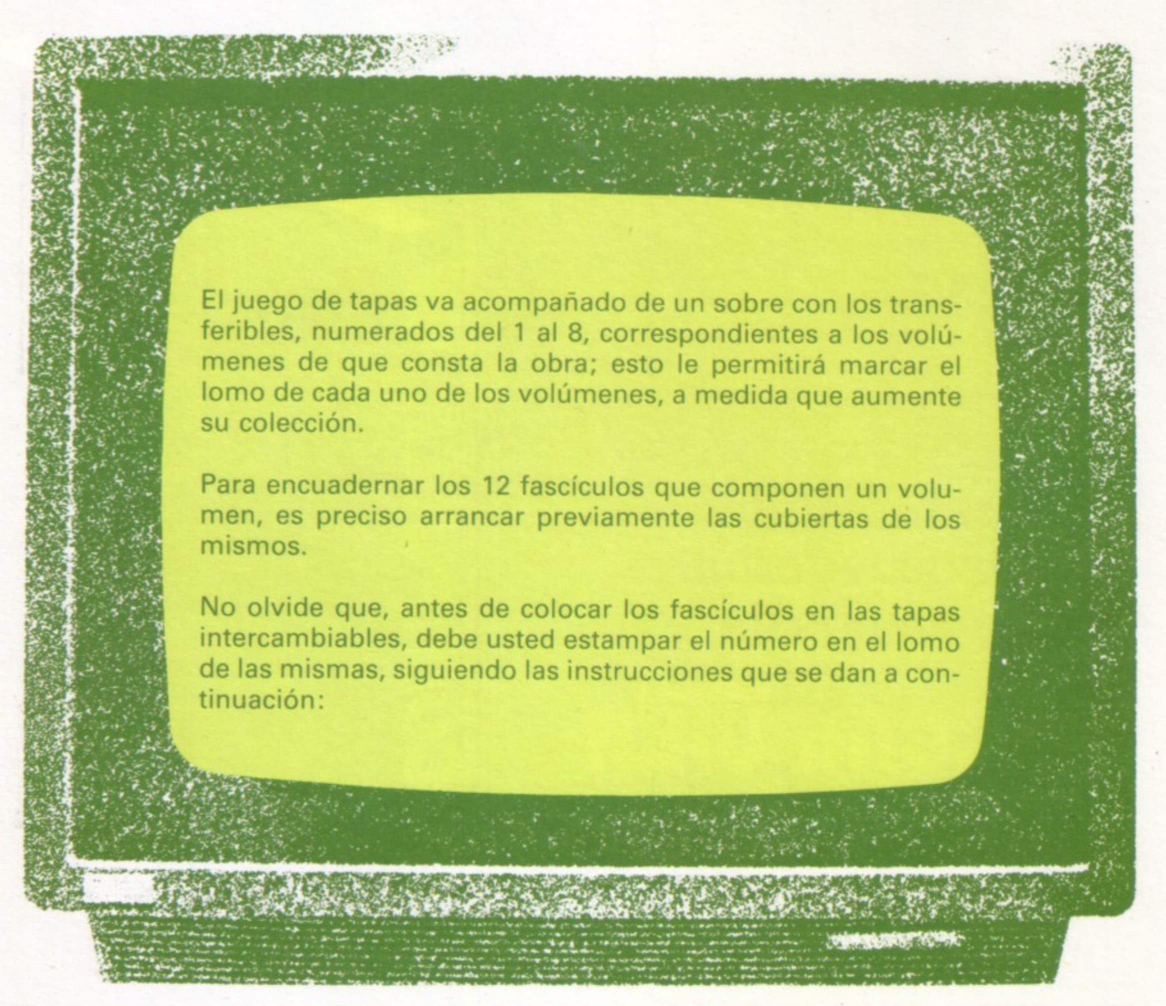


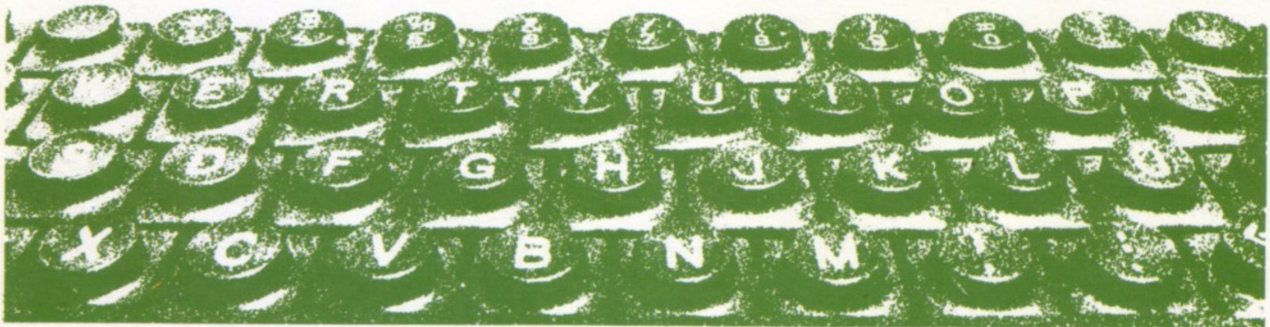
El Manchester Mark I
Por haber logrado ejecutar un programa con pleno éxito en junio de 1948, se puede decir que el Manchester Mark I fue el primer ordenador del mundo con programa almacenado. Ferranti, entonces una empresa local, recibió el encargo de fabricar una versión comercial del ordenador, que salió al mercado a principios de 1951

Cortesía de la Universidad de Manchester

mi COMPUTER

Con el próximo fascículo se pondrán a la venta las tapas correspondientes al segundo volumen





1 **Desprenda la hojita de protección y aplique el transferible en el lomo de la cubierta, haciendo coincidir los ángulos de referencia con los del recuadro del lomo.**

2 **Con un bolígrafo o un objeto de punta roma, repase varias veces el número, presionando como si quisiera borrarlo por completo.**

3 **Retire con cuidado y comprobará que el número ya está impreso en la cubierta. Cúbralo con la hojita de protección y repita la operación anterior con un objeto liso y redondeado, a fin de asegurar una perfecta y total adherencia.**

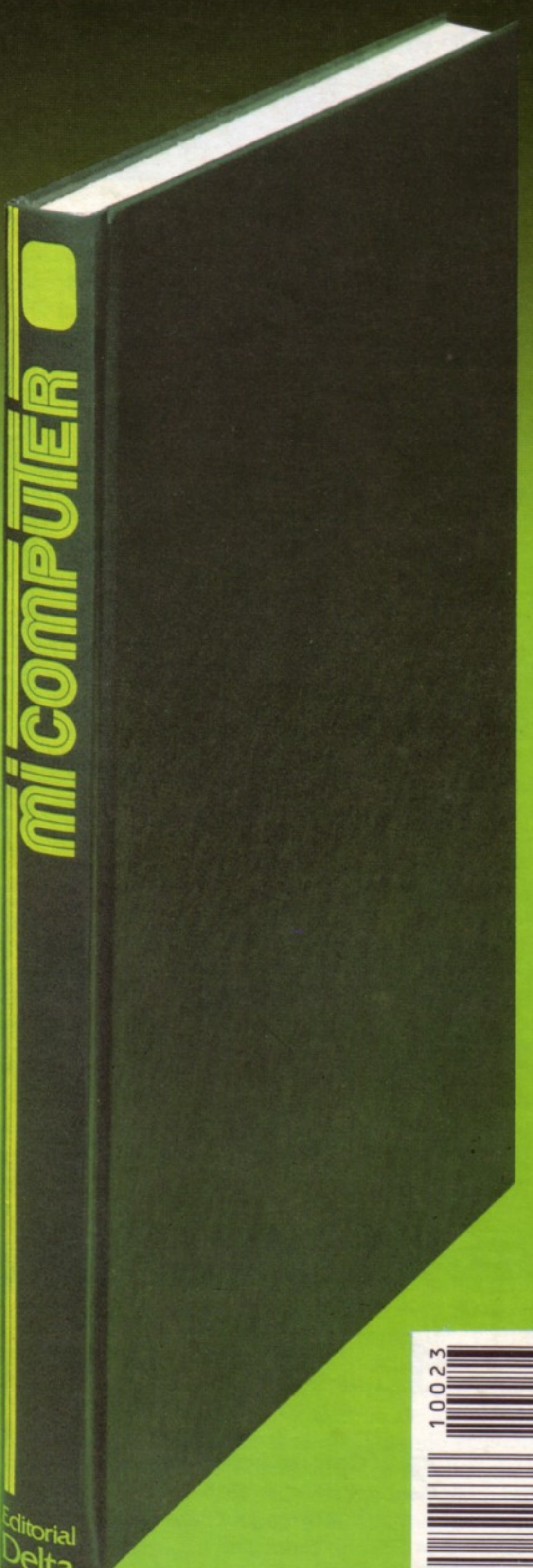
Cada sobre de transferibles contiene una serie completa de números, del 1 al 8, para fijar a los lomos de los volúmenes. Ya que en cada volumen sólo aplicará el número correspondiente, puede utilizar los restantes para hacer una prueba preliminar.

Editorial Delta

Con el próximo número se ponen a la venta las tapas intercambiables para encuadernar 12 fascículos de

mi computer

Cada juego de tapas va acompañado de una colección de transferibles, para que usted mismo pueda colocar en cada lomo el número de tomo que corresponda





9 788485 822836
10023